FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Framework para o Desenvolvimento de Aplicações Orientadas à Agricultura de Precisão Baseadas em Redes De Sensores e Atuadores Sem Fios

Ricardo Jorge da Silva Gama Nogueira

DISSERTAÇÃO

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Ricardo Santos Morla

30 de Julho de 2014

Framework para o Desenvolvimento de Aplicações Orientadas à Agricultura de Precisão Baseadas em Redes De Sensores e Atuadores Sem Fios

Ricardo Jorge da Silva Gama Nogueira

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: João António Correia Lopes

Arguente: Pedro Miguel Alves Sobral

Vogal: Ricardo Santos Morla

---

30 de Julho de 2014

Resumo

Historicamente, a agricultura apresenta-se como uma das mais antigas e importantes atividades económicas, tendo sofrido uma grande expansão nos últimos séculos devido à evolução tecnológica que permite a cada passo tornar a agricultura uma atividade mais eficiente. Desta forma, o conceito Agricultura de Precisão surge com a utilização de equipamentos de alta tecnologia para produzir colheitas em maior quantidade, melhor qualidade e mais rapidamente. Os sensores e atuadores assumem-se como parte destas tecnologias uma vez que permitem a monitorização e o controlo em níveis ótimos dos fatores ambientais que influenciam o crescimento das colheitas.

Tendo em conta a importância destas tecnologias, esta dissertação foca-se no desenvolvimento de uma Framework para o desenvolvimento de aplicações baseada em Redes de Sensores e Atuadores Sem Fios (RSASF). As aplicações construídas através da Framework devem ser genéricas ao ponto de poderem ser utilizadas em todos os tipos de culturas agrícolas, oferecendo no entanto uma grande variedade de funcionalidades. A Framework pretende superar alguns problemas típicos do desenvolvimento deste tipo de aplicações, que estão normalmente associados à complexidade tecnológica dos dispositivos e sistemas que as suportam. Além disso, a Framework foi desenvolvida com o objetivo de oferecer uma grande flexibilidade e extensibilidade, no sentido de permitir integrar facilmente novos equipamentos e funcionalidades nas aplicações criadas, de forma a se destacar do Estado da Arte das aplicações baseadas em RSASF.

A Framework foi desenvolvida a partir da PlugSense Framework, uma framework que permite gerar aplicações finais baseadas em Redes de Sensores Sem Fios a partir de uma aplicação genérica, oferecendo ferramentas para integrar novos sensores de forma ágil e incluí-las nas aplicações existentes de forma flexível. O processo de desenvolvimento no âmbito desta tese começou pela integração de um novo tipo de sensor na Framework que permite recolher 44 dados ambientais, com o objetivo de oferecer às aplicações geradas uma elevada capacidade de monitorização do ambiente de crescimento de colheitas.

Foram feitas alterações a nível da estrutura da aplicação genérica que a Framework permite gerar, de forma a esta poder representar as entidades presentes no contexto da agricultura, bem como acrescentar funcionalidades úteis neste contexto, nomeadamente a visualização de modelos de prevenção de doenças e pestes em tempo real com base nos dados recolhidos pelos sensores e a georreferenciação das parcelas de terreno.

Uma vez que a versão PlugSense Framework não suporta atuadores, foram desenvolvidos mecanismos e estruturas que permitem a integração ágil de atuadores na Framework, bem como a capacidade oferecer uma interação remota entre os utilizadores das aplicações e os dispositivos.

Por fim, com a colaboração da Associação de Produtores Agrícolas da Sobrena, criou-se um caso de uso onde foi possível efetuar a validação da Framework e dos objetivos propostos. Assim, a partir da Framework desenvolvida foi criada uma aplicação final para a utilização desta associação de acordo com os seus requisitos, provando a utilidade desta ferramenta.

Abstract

Historically, agriculture presents itself as one of the oldest and most important economic activities, having undergone a major expansion in the last century due to technological developments that allows agriculture to be a more efficient activity in each step. Thus, the concept of Precision Agriculture arises with the use of high-tech equipment to produce crops in greater quantity, better quality and faster. The sensors and actuators are assumed as part of these technologies because they enable the monitoring and control at optimal levels of environmental factors that influence crop growth.

Given the importance of these technologies, this dissertation focuses on the development of a Framework for the development of application based on Wireless Sensor and Actuator Networks (WSAN) applications. The applications built through the Framework are expected to be generic to the point that they can be used in all types of crops, while still offering a wide variety of features and a low usability. The Framework aims to overcome some typical problems of the development of such applications, which are usually associated with the technological complexity of the devices and systems that support them. Furthermore, the Framework was developed with the goal of offering great flexibility and extensibility conditions, and provide mechanisms to easily integrate new devices and functionalities to existent applications, in order to highlight the Framework from the state of the art of applications based on WSAN.

The Framework was developed from the PlugSense Framework, a framework to generate final applications based on Wireless Sensor Networks from a generic application, offering tools to integrate new sensors expeditiously and include them in existing applications flexibly. The development process in the context of this thesis began by integrating a new type of sensor that collects 44 environmental data, with the goal of providing a high capacity to monitor crop growth environment to the generated applications.

In order to represent the entities present in the context of agriculture, changes in the structure of general application were made, as well as adding features useful in this context, namely the visualization of models of disease and pest prevention in real time based on data collected by sensors and geo-referencing of parcels of land.

Since the original version of the Framework does not support actuators, mechanisms and structures that enable swift integration of actuators in the Framework as well as the ability to provide a remote interaction between users of the applications and devices have been developed.

Finally, in collaboration with the Agricultural Producers Association of Sobrena, a case study was created in which was possible to perform the validation of the Framework and the proposed objectives. Thus, according to the requirements of this association an ultimate application was created from the developed Framework, proving the usefulness of this tool.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais António e Luísa Nogueira, por terem sempre acreditado incondicionalmente nas minhas capacidades e oferecido todas as condições necessárias para que o meu percurso académico se realizasse.

Agradeço também ao meu irmão Carlos Nogueira por ter sido um exemplo constante de ambição e determinação, e me ter trazido inspirição em tantos momentos de incerteza.

Por fim, agradeço ao Prof. Ricardo Morla e ao Eng. Marco Alves por todo o apoio, disponibilidade e orientação oferecidos durante a realização desta tese, permitindo que concluísse a última etapa do meu percurso académico com um enorme sentimento de satisfação e realização.

Ricardo Nogueira

Conteúdo

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

Abreviaturas e Símbolos

AP	Agricultura de Precisão
RSSF	Redes de Sensores Sem Fios
RSASF	Redes de Sensores e Atuadores Sem Fios
ISM	*Industrial, Scientifical and Medical*
RF	*Radio Frequency*
PSF	PlugSense Framework
UG	Universal Gateway
APAS	Associação dos Produtores Agrícolas da Sobrena
DL	DataLogger
WL	WeatherLink

Capítulo 1

Introdução

A agricultura é uma actividade básica, imprescindível para a satisfação de inúmeras necessidades humanas (alimentares, agasalho, energia, etc.), sendo uma das mais antigas actividades económicas. Ao longo do tempo, a agricultura foi evoluindo e modificando-se muito lentamente, mas no último século sofreu uma enorme transformação. Primeiro, com a revolução industrial e a adopção da potência da máquina, a agricultura mecanizou-se. Mais tarde, com a inclusão dos avanços no domínio da óleo-dinâmica, sofisticaram-se as máquinas que passaram a ter mecanismos de assistência no comando e controlo. Mais recentemente, assistimos à introdução de variados complementos dos sistemas anteriores, suportados por sensores electrónicos, que possibilitaram a criação de automatismos. Finalmente, ao longo da última década, temos vindo a assistir à crescente adopção e integração com as tecnologias de informação.

O termo Agricultura de Precisão (AP) está normalmente associado à utilização de equipamentos de alta tecnologia (seja *hardware* ou *software*) para avaliar, ou monitorizar, as condições numa determinada parcela de terreno, aplicando depois os diversos fatores de produção (sementes, fertilizantes, fitofármacos, reguladores de crescimento, água, etc.) [CdS09], sendo estes aplicados no terreno por componentes mecânicos genericamente denominados atuadores.

Este estudo pretende levar a cabo a conceção e desenvolvimento de uma framework (a partir de uma framework existente) que permite construir aplicações dirigidas à AP. As aplicações construídas através da Framework poderão, entre outras funcionalidades, incluir a utilização e interação de um conjunto de sensores eletrónicos e atuadores, tirando partido das várias tecnologias de comunicação sem fios existentes.

A Dissertação foi realizada em ambiente empresarial na Freedom Grow, localizada em Matosinhos. Esta empresa é detentora do software PlugSense Framework, que foi utilizada como base do desenvolvimento da Framework, e que será apresentada com mais detalhe na Secção 2.5.

Neste capítulo serão abordados alguns conceitos relevantes na AP, a motivação para a realização desta dissertação, e as tecnologias envolvidas no desenvolvimento da Framework.

1.1 Contexto

Os sistemas de sensores utilizados na agricultura foram já apelidados de vários maneiras, como Precision Agriculture (PA), Smart Agriculture, Variable Rate Technology (VRT), Precision Farming, Global Positioning System (GPS) Agriculture, Farming by Inch, Information-Intensive Agriculture, Site Specific Crop Management etc. [Sri06], sendo no entanto o conceito por detrás destas terminologias exatamente o mesmo.

A base da AP está na variabilidade espacial e temporal do solo e colheitas no terreno. Antes de agricultura se ter mecanizado, o pequeno tamanho dos terrenos permitia que os agricultures variassem o tratamento manualmente. No entanto, o alargamento dos terrenos e a intensiva mecanização tornou cada vez mais difícil de tomar conta das variações nos terrenos sem um desenvolvimento revolucionário em tecnologias [Sta00].

Segundo [ZWW02], as variações espaciais e temporais que têm influência significativa na produção agrícola podem ser categorizada em seis grupos:

- **Variação da quantidade de produção** — Distribuições de rendimento histórico e do presente.
- **Variações do terreno** — Topografia do solo: elevação, declive, etc.
- **Variações do solo** — Fertilidade do solo: N, P, K, Ca, Mg, C, Fe, Mn, Zn, Cu; fertilidade do solo prevista pela aplicação de estrume; propriedades físicas e de textura do solo: densidade, resistência mecânica, humidade, etc.
- **Variações da planta** — Densidade de culturas; altura da colheita; *stress* de nutrientes das culturas para N, P, K, Ca, Mg, C, Fe, Mn, Zn e Cu; *stress* hídrico; propriedades biofísicas - índice de área foliar (LAI), radiação fotossinteticamente ativa e biomassa; teor de clorofila nas folhas de culturas; e qualidade de grãos da cultura.
- **Variações de fatores anómalos** — Infestação de ervas daninhas; infestação de insetos; infestação de nematóides; infestação de doenças; danos provocados pelo vento e feno.

Gerir a variabilidade na agricultura pode ser conseguida de duas formas: a abordagem baseada em mapeamento e a abordagem baseada em sensores [ZWW02]. Os parâmetros ambientais de um terreno agrícola podem ser vistos como uma rede de informação espacial e tridimensional. Desenvolver métodos para recolher, processar, integrar e aplicar informação ambiental é o objetivo da tecnologia de informação ambiental contemporânea e da ciência agrícola contemporânea. Devido às regiões agrícolas serem dispersas e por isso as condições ambientais e terrestres variarem significativamente, métodos para recolher de uma forma precisa e rápida informação variada que afete o crescimento das colheiras são um dos principais problemas para a investigação agrícola e ambiental [STPM05]. A tecnologia de Rede(s) de Sensores Sem Fios (RSSF) surgiu como o meio técnico para resolver este problema.

As RSSF apresentam-se como uma parte essêncial de todos os sistemas que precisam de efetuar medições de atributos físicos ou ambientais [WZW06]. Uma RSSF num ambiente agrícola contém sensores integrados implantados numa área agrícola. Estes sensores cooperam entre si para recolher e monitorizar informações climatéricas e do solo em tempo real. A informação é processada inteligentemente por um sistema integrado. Além disso, a informação é transmitida para um centro de decisão e diagnóstico por uma rede de comunicação sem fios auto-organizada, que fornece monitorização e gestão remota do ambiente agrícola [YWHZ13]. No capítulo 2 a arquitetura e características técnicas das RSSFs serão abordadas com mais detalhe.

As Redes de Sensores e Atuadores Sem Fios (RSASF) são uma variante das RSSF que têm um componente adicional, o atuador. A inclusão de atuadores aumenta a capacidade das RSSF de apenas monitorização a controlo. Além disso, sensores sem fios e atuadores são necessários para recolher a informação necessária para reagir a diversas situações. O suporte à decisão impõe o requisito de ter informação processada em vez de dados brutos recolhidos pelos sensores [aSYN+10]. Esta nova abordagem tem vindo a oferecer um enorme contributo no desenvolvimento de aplicações inteligentes no domínio da computação ubíqua como campus inteligentes [uRAS08], hospitais inteligentes [FG06], etc. As RSASF têm sido utilizadas na AP para gerir irrigação, cultivação, fertilização, etc. No Capítulo 3 são revistas alguns exemplos de aplicações de RSASF.

1.2 Motivação

Existe um enorme potencial na combinação de RSSF e aquisição de dados em tempo real para melhorar a gestão agrícola [The03]. A gestão eficiente do processo agrícola requer sistemas de aquisição de dados do ambiente em que se insere, implicando a utilização de vários sensores específicos para estimar o crescimento de colheitas. Apesar deste requisito, a maior parte dos campos e estufas agrícolas não integram ou exploram todas as capacidades dos sistemas de controlo digitais modernos, devido a fatores como o custo e dificuldade de operar e integrar as soluções disponíveis [MVS05].

Tendo em conta a utilização de atuadores nas aplicações para AP, a irrigação de culturas agrícolas é uma das mais importantes áreas da agricultura e a água apresenta-se como um recurso escasso que motiva a necessidade de irrigação controlada [aSYN+10]. Alguns métodos já estão a ser utilizados, como irrigação por gota-a-gota, que oferecem um fornecimento de água controlado, mas é necessário o auxílio de *software* para efetuar decisões como a quantidade e o modo de irrigação necessárias. Este tipo de irrigação inteligente é essencial na AP [uRS09].

Algumas das principais preocupações no desenvolvimento de sistemas tecnológicos para a AP são atributos como modularidade, flexibilidade, extensibilidade, escalabilidade, programabilidade, integração configurável e interoperabilidade [SMK13]. As RSSF deverão ser escaláveis e permitir a adaptação dinâmica a alterações na topologia e densidade dos nodos. O ciclo de vida é extremamento crítico para a maioria das aplicações, sendo o consumo de energia o principal fator limitante dos nodos, que deverão ser auto-sustentáveis [PH05].

No desenvolvimento de sistemas de aquisição de dados de um ambiente agrícola, as seguintes características são essencias para assegurar o seu bom funcionamento:

- **Mobilidade dos dispositivos** — para superar restrições das actividades agrícolas;
- **Registo e conexão automática dos dispositivos na rede** — para facilitar a instação e alteração dos equipamentos no terreno;
- **Capacidade de encaminhamento** — para garantir o acesso a dispositivos no caso de falha temporária ou permanente em alguma parte da rede;
- **Gestão de consumo energético** — para fornecer monitorização e optimização dos custos e recursos energéticos.

Para satisfazer tais requisitos, sensores sem-fios, atuadores e as redes que os incluem são essenciais ao desenvolvimento de sistemas de aquisição de contexto, apresentando os dados recolhidos a sistemas de suporte à decisão e oferecendo assim um controlo do ambiente baseado em decisões [MVS05].

Considerando estes fatores, pretende-se desenvolver uma Framework capaz de gerar aplicações genéricas dirigidas à AP baseada em RSASF. Pretende-se que a Framework se imponha como uma ferramenta valiosa no desenvolvimento deste tipo de aplicações, permitindo facilitar ou mesmo evitar algumas fases do processo de conceção, que têm normalmente um período de desenvolvimento muito elevado.

Estas aplicações devem conter uma estrutura e lógica acessíveis ao contexto da agricultura. Apesar da estrutura genérica, as aplicações deverão, no entanto, conter um conjunto de funcionalidades que possam ser úteis nas várias áreas da agricultura.

A partir da aplicação genérica, será possível construir aplicações dirigidas a uma área da agricultura em particular, adaptando os seus conteúdos e acrescentando novas funcionalidades focadas nas necessidades dessa atividade agrícola e nas funções que os utilizadores irão desempenhar.

Uma das características que irá destacar a Framework do presente Estado da Arte é a abstração das necessidades e potenciais complicações técnicas da implementação do sistema no cenário de utilização, como a variedade e especificidade dos dispositivos necessários, variação topográfica dos terrenos ou tecnologias de comunicação envolvidas.

Neste sentido, a Framework deverá ser flexível ao ponto de poderem ser facilmente integrados novos sensores e atuadores nas aplicações e apresentar uma arquitetura robusta capaz de se adaptar a qualquer cenário de utilização.

No capítulo 3 serão aprofundados o objetivos e conceitos da aplicação.

1.3 Caso de uso

No início do processo da Dissertação foi estabelecida uma parceria com a Associação dos Produtores Agrícolas da Sobrena[1], com o objetivo de construir uma aplicação através da Framework

[1] www.apas.com.pt

adequada às suas necessidades. Esta parceria permitiu não só validar a usabilidade das aplicações criadas através da Framework num caso real de utilização, como permitiu expandir as suas funcionalidades.

A APAS dispõe de um conjunto de estações meteorológicas que incorporam um variado leque de sensores. Através de folhas de cálculo e com base nos dados recolhidos das estações, são executados modelos de cálculo de risco e prevenção das doenças mais comuns nas culturas de peras e maçãs em Portugal. Posteriormente, estas folhas de cálculo são fornecidas em forma de serviço a agricultores que estejam interessados nesta informação para poderem aplicar as ações necessárias a prevenir que as suas culturas sejam prejudicadas por estas doenças ou pragas.

1.4 Estrutura do Relatório

No próximo capítulo serão abordadas as tecnologias utilizadas no desenvolvimento da Framework, bem como alguns conceitos fundamentais à compreensão do problema e da solução proposta.

No Capítulo 3, será apresentada a revisão bibliográfica efetuada no sentido de estudar o Estado da Arte das aplicações baseadas em RSASF. Será feita uma análise e reflexão sobre as aplicações estudadas de forma a obter uma maior definição do problema bem como refinar a perspetiva de solução.

No Capítulo 4 é feita uma descrição pormenorizada do processo de desenvolvimento da Framework, seguindo para o Capítulo 5 onde será abordada fase de validação, e que inclui o resultado do caso de uso da aplicação criada para a APAS através da Framework desenvolvida.

Por fim, o documento termina com o Capítulo 6 reservado às conclusões obtidas nesta dissertação, bem como algumas perspetivas de trabalho futuro.

Capítulo 2

Revisão tecnológica

De seguida é feita uma revisão das tecnologias utilizadas no desenvolvimento da Framework, bem como uma abordagem a conceitos essenciais no contexto das RSASF.

2.1 Sensores

A utilização de sensores está-se a tornar possível em quase todas as áreas da vida devido ao avanço da tecnologia e à diminuição do seu tamanho. Um sensor é um dispositivo com capacidade de medir atributos físicos e convertê-los em sinais para observação. Assim, os sensores acumulam informação que caracteriza ambientes e são utilizados para identificar pessoas, localizações e objetos, e os seus estados são adquiridos em contexto [ST94, ADB+99].

A aquisição de contexto fornece assim uma contribuição valiosa em modelar situações de domínios que apresentam variações temporais e de atributos, sendo a agricultura um desses domínios. Devido à variedade dos fatores que são relevantes no domínio da agricultura, que variam ainda mais com a especificidade de cada área da agricultura, é necessária a utilização de sensores que efetuem medições a três níveis distintos: solo, ambiente e planta/folha. Na Figura 2.1 são apresentados sensores utilizados frequentemente na AP, divididos pelos níveis acabados de referir.

Os sensores são dispositivos que têm a capacidade de converter dados ambientais em sinais, pelo que as RSSF necessitam de equipamentos com capacidade de processamento e encaminhamento para formar uma rede. Assim, existem dispositivos inteligentes que têm o objetivo de comunicar diretamente com os sensores e proceder ao encaminhamento dos dados através de tecnologias de comunicação sem fios. Estes dispositivos são normalmente denominados por nodos sensor ou *motes*, podendo alguns destes equipamentos ter sensores incorporados, simplificando a topologia das redes. Os *motes* são a unidade básica das RSSF e contêm 4 módulos, nomeadamente os módulos Sensor/Atuador, Comunicação, Processamento e Energia. Como opção podem conter memória externa no caso de ser necessário armazenamento local para tomada de decisões. O seu *design* requer algumas considerações como conservação de energia, escabilidade e tamanho.

Soil

S. no.	Sensors	Temperature	Moisture	Dielectric permittivity	Rain/ water flow	Water level	Conductivity	Salinity	References
1	Hydra probe II soil sensor	✓	✓	✓	✓	✓	✓	✓	www.stevenswater.com
2	Pogo portable soil sensor	✓	✓	✓	✓	–	✓	–	www.stevenswater.com
3	MP406 Soil moisture sensor	✓	✓	✓	–	–	–	–	www.ictinternational.com.au
4	ECH2O soil moisture sensor	✓	✓	✓	–	✓	✓	–	www.ictinternational.com.au
5	EC sensor (EC250)	✓	✓	–	✓	–	✓	✓	www.stevenswater.com/catalog/products/water_quality_sensors/manual
6	ECRN-50 low-REC rain gauge	–	–	–	✓	–	–	–	http://www.decagon.com
7	ECRN-100 high-REC rain gauge	–	–	–	✓	–	–	–	http://www.decagon.com
8	Tipping buket rain gage	–	–	–	✓	–	–	–	www.stevenswater.com
9	107-L temperature Sensor (BetaTherm 100K6A1B Thermistor)	✓	–	–	–	–	–	–	http://www.campbellsci.com/107-l

Leave/plant

S. no.	Sensors	Photosynthesis	Moisture	Hydrogen	Wetness	CO_2	Temperature	References
1	237 leaf wetness sensor	–	✓	–	✓	–	✓	http://www.campbellsci.com
2	LW100, leaf wetness sensor	–	✓	–	✓	–	✓	http://www.globalw.com LW100B.pdf
3	SenseH2™ hydrogen sensor	–	–	✓	✓	✓	✓	http://www.NTMSENSORS.com
4	Leaf wetness sensor	–	✓	–	–	–	–	http://www.decagon.com
5	YSI 6025 chlorophyll sensor	✓	–	–	–	–	–	http://www.ysi.com ysi_6025.pdf
6	Field scout CM1000TM	✓	–	–	–	–	–	http://www.specmeters.com/pdf/2950FS.pdf
7	TT4 multi-sensor thermocouple	–	✓	–	–	–	✓	www.ictinternational.com.au/thermocouple.htm
8	LT-2 M (leaf temperature sensor)	–	–	–	–	–	✓	http://www.solfranc.com LT-2 M Leaflet SOLFRANC.pdf
9	TPS-2 portable photosynthesis	✓	✓	–	✓	✓	✓	www.ppsystems.com/Literature/EDSTPS2_System.pdf
10	PTM-48A photosynthesis monitor	✓	✓	–	✓	✓	✓	http://phyto-sensor.com/PTM-48A
11	CI-340 hand-held photosynthesis	✓	✓	✓	✓	✓	✓	http://www.solfranc.com CI-340_hand-held_photosynthesis_solfranc_ENG.pdf
12	107-L temperature Sensor (BetaTherm 100K6A1B thermistor)	✓	–	–	–	–	–	http://www.campbellsci.com/107-l

Weather

S. no.	Sensors	Temperature	Humidity	Atmospheric pressure	Wind speed	Wind direction	References
1	CM-100 compact Weather sensor	✓	✓	✓	✓	✓	www.stevenswater.com
2	Met station one (MSO)	✓	✓	✓	✓	✓	www.stevenswater.com
3	XFAM-115KPASR	✓	✓	✓	–	–	http://www.pewatron.com 100-31-102-006-EH-0110.pdf
4	HMP45C (Vaisala's HUMICAP® H-chip)	✓	✓	✓	–	–	http://www.campbellsci.com HMP45C Temperature and Relative Humidity Probe
5	SHT71 (Humidity and temperature sensor)	✓	✓	✓	–	–	http://www.sensirion.com/humidity
6	SHT75 (Humidity and temperature sensor)	✓	✓	✓	–	–	http://www.sensirion.com/humidity

Figura 2.1: Sensores utilizados no domínio da agricultura [AIS11]

A escolha de um *mote* tem em consideração os requsitos do problema e da aplicação com mais impacto, e o seu padrão de distribuição. O processador/microcontrolador, memória, frequência de banda, alcance de transmissão e tamanho são algumas das características que fazem a diferença na escolha de um *mote*. Na Figura 2.2 é apresentada uma lista de vários *motes* disponíveis no mercado.

Analisando a tabela da Figura 2.2, a maioria dos *motes* inclui o microcontrolador Atmega128L, transdutores ZigBee (CC2420) ou RF (CC1000) funcionando numa frequeência de banda Industrial, Médica e Científica (ISM), e com sensores incorporados que recolhem informação climatérica. Isto verfica-se devido aos atributos especializadas do microcontrolador, licensa gratuita de banda ISM e capacidade de ter sensores incorporados serem propriedades muito úteis para a maioria dos problemas/sistemas [AIS11].

Feature	MICAz	MICA2DOT	MICA2	Imote2	TelosB	IRIS	Cricket
Microcontroller	ATmega128L	ATmega128L	ATmega128L	Marvell/ XScalePXA271	TIMSP430	ATmega128L	ATmega128L
Clock Speed (MHz)	7.373	4.0	7.373	13–416	6.717	7.373	4.0
Bus Size (bits)	8	8	8	32	16	8	8
Memory (bytes)	EEPROM:4 K	EEPROM:4 K	EEPROM:4 K	SRAM:256 K SDRAM:32 M	RAM:10 K EEPROM: 16 K	EEPROM: 4 K RAM:8 K	EEPROM:4 K
Memory(flash) (bytes)	Program: 128 K	Program: 128 K	Program: 128 K	Programmable	Program: 48 K	Program: 128 K	Program: 128 K
	Serial: 512 K	Serial: 512 K	Serial: 512 K	Flash: 32 M	Serial: 1024 K	Serial: 512 K	Serial: 512 K
Size (mm)	58×32×7	58×32×7	58×32×7	36×48×9	65×31×6	58×32×7	58×32×7
Weight (gm)*	18	18	18	12	23	18	18
Battery	2×AA	3 V Coin cell CR2354	2×AA	3×AA	2×AA	2×AA	2×AA
External power	2.7 V–3.3 V	3 V	2.7 V–3.3 V	3.2 V–4.5 V	2.7 V–3.3 V	2.7 V–3.3 V	2.7 V–3.3 V
Active mode power (mW) consumption	24@3 V	15@3 V	24@3 V	44@13 MHz 570@416 MHz	10@3 V	24@3 V	24@3 V
Sleep mode power (μW) consumption	75	75	75	100	8	24	75
Available sensors	Light, temperature, humidity, barometric pressure, accelerometer, GPS, RH, acoustic, video sensor, microphone, sounder, magnetometer	Light, Temperature, Accelerometer	Light, temperature, humidity, barometric pressure, accelerometer, GPS, RH, acoustic, video sensor, microphone, sounder, magnetometer	Light, Temperature, Humidity, Accelerometer	Light, temperature, humidity	Light, temperature, RH, barometric pressure, acceleration/ seismic, acoustic, magnetic and video	Light, temperature, humidity, barometric pressure, accelerometer, GPS, RH, acoustic, ultrasonic, video sensor, microphone, sounder, magnetometer
User interface	3 LEDs	1 LED**	3 LEDs**	USB client/ host	USB	3 LEDs	3 LEDs**
Expansion connector	51-pin	19-pin	51-pin	40-pin and 20-pin	6-pin and 10-pin	51-pin	51-pin
Serial communication	UART	UART	UART	UART 3x	UART	UART	UART
Other interfaces	Digital I/O, I2C, SPI	DIO (8 channels)	DIO, I2C, SPI	GPIOs, I2C, DI0, JTAG, SPI2x, I2S, AC97, Camera	Digital I/O, I2C, SPI	Digital I/O, I2C, SPI	DIO, I2C, SPI
Transceiver chip	CC2420	CC1000	CC1000	CC2420	CC2420	Atmel RF230	CC1000
Frequency band (Mhz) ISM band	2400–2483.5	868/916	868/916	2400–2483.5	2400–2483.5	2400–2480	868/916
Number of channels	In steps of 1 MHz***	4/50	4/50	In steps of 5 MHz	In steps of 1 MHz***	In steps of 1 MHz***	4/50
Data Rate	250 Kbps	38.4 K Baud	38.4 K Baud	250 Kbps	250 Kbps	250 Kbps	38.4 K Baud
RF Transmit power (dBm)	−24 to 0	−20 to +5	−20 to +5	−24 to 0	−24 to 0	+3	−20 to +5
Receive (dBm) Sensitivity	−94	−98	−98	−94	−94	−101	−98

*Without batteries, **User programmable, and ***Programmable.

Figura 2.2: Comparação entre alguns nodos de ligação disponíveis [AIS11]

O microcontrolador desempenha um papel essencial pois fornece capacidade de processamento para tomada de decisões locais, agregação de dados e gestão de consumos energéticos através de modos *sleep*, etc. O Atmega128L é o microcontrolador mais comum devido ao seu baixo consumo de energia, vários modos *sleep*, memória *flash*, armazenamento orientado a bytes eficiente e compatibilidade com praticamente todos os códigos TinyOS.

2.2 Atuadores

Um atuador é um mecanismo pelo qual um sistema de controlo atua sobre o ambiente. O sistema de controlo pode ser simples, como um sistema elétrico ou mecânico fixo, baseado em *software*, humano, ou outro tipo de *input*.

A utilização de atuadores é muito frequente na indústria para automação de válvulas industriais, que são essenciais no controle do processo de automatização. Podem ser utilizadas em indústrias como mineração, estações de tratamento de águas residuais, refinarias, processos nucleares ou oleodutos. Dependendo do seu tipo de alimentação, os atuadores podem ser classificados como atuadores pneumáticos, hidráulicos ou elétricos.

Na indústria agrícola, os atuadores são muito utilizados para controlar válvulas de irrigação, que é um mecanismo presente em praticamente todos os tipos de culturas. No entanto, existem culturas onde a cultivação e crescimento das colheitas é feito em estufas, onde o controlo do ambiente é um fator decisivo para obter níveis ótimos para o crescimento das colheitas. Neste contexto, a utilização de atuadores que controlam outro tipo de dispositivos são muito úteis, como aplicador de fertilizantes, garrafas de enriquecimento de CO2, cortinas mecanizadas e luz artificial, ares condicionados, janelas mecanizadas ou válvulas de vaporização de água.

O Fieldbus é uma família de protocolos de redes de computadores industriais utilizados no controlo distribuído em tempo real, padronizado como IEC 61158. Esta tecnologia é cada vez mais utilizada para a transmissão de dados em aplicações de automação de processos, onde os atuadores elétricos podem ser equipados com todas as interfaces Fieldbus comuns em automação de processos. Alguns dos protocolos Fieldbus mais conhecidos são Bitbus, EtherCAT, Interbus, LonWorks e Modbus.

2.3 Tecnologias de comunicação

2.3.1 Comunicação sem fios

Tecnologias de comunicação sem fios como ZigBee, Bluetooth, Wibree e WiFi são frequentemente utilizadas em trabalhos de pesquisa baseados em RSASF [ZlYmZ+07]. Estas tecnologias têm diferentes propriedades e capacidades, estando presente na Tabela 2.1 uma breve comparação entre elas.

A tecnologia de comunicação sem fios ZigBee tem vindo a ser utilizada em maior escala devido ao seu baixo custo e baixo consumo de energia. Foi introduzida em Maio de 2003 e funciona na banda ISM, isto é 2.4 GHz globalmente. Existem 16 canais ZigBee de largura de banda 5 MHz em cada banda de 2.4 GHz.

Tabela 2.1: Comparação entre tecnologias de comunicação sem fios [uRAS08]

	ZigBee	Bluetooth	WiBree	WiFi
Frequency band	2.4 Ghz	2.4 Ghz	2.4 Ghz	2.4 Ghz
Range	30 m – 1.6 km	30–300 ft	Up to 10 ft	100–150 ft
Data rate	250 kbps	1 Mbps	1 Mbps	11–54 Mbps
Power consumption	Low	Medium	Low	High
Cost	Low	Low	Low	High
Modulation/protocol	DSSS, CSMA/CA	FHSS	FHSS	DSSS/CCK, OFDM
Security	128 bit	64 or 128 bit	128 bit	128 bit

2.3.2 Protocolo Modbus

O Modbus é um protocolo da camada de aplicação, posicionado no nível 7 do modelo OSI, que proporciona a comunicação sobre o padrão *master/slave*, entre dispositivos ligados em diferentes redes [Org12].

O modelo de dados deste protocolo baseia-se em 4 tabelas principais(ou *arrays*), apresentados na tabela 2.2, que apresentam características e modos de enderaçamento diferentes, para serem utilizados em diferentes situações.

Tabela 2.2: Modelo de dados no protocolo Modbus

Tabelas primárias	Tamanho dos objetos	Tipo de acesso
Discretes Input	1 bit	Read-Only
Coils	1 bit	Read-Write
Input Registers	16 bit	Read-Only
Holding Registers	16 bit	Read-Write

Nos sistemas baseados em Modbus existe um dispositivo *master* que efetua pedidos a vários dispositivos denominados *slave*. Os pedidos no protocolo Modbos são encapsulados em funções, que são especificadas por um código único e apresentam formatos de mensagens específicos. Algumas das funções mais relevantes do protocolo ModBus são apresentadas na tabela 2.3.

Tabela 2.3: Funções do protocolo Modbus

Tamanho do acesso	Tipo de objeto	Função	Código (hex)
Bit access	Physical Discrete Inputs	Read Discrete Inputs	02
		Read Coils	01
	Interanl Bits or Physical Coils	Write Single Coil	05
		Write Multiple Coils	0F
16 bit access	Physical Input Registers	Read Input Register	04
		Read Holding Registers	03
	Internal Registers or Physical Output Registers	Write Single Register	06
		Write Multiple Registers	10
		Read/Write Multiple Registers	17

Existem diferentes formas de efetuar a comunicação entre o dispositivo *master* e os *slaves*. Numa abordagem, o *master* está ligado fisicamente aos *slaves* através de redes baseadas em RS-485 ou RS-232, havendo as versões Modbus RTU e Modbus ASCII, onde os endereços e valores são representados em formato binário e caracteres ASCII, respetivamente. Alternativamente, o *master* pode efetuar os pedidos encapsuladas em quadros TCP, e através do protocolo TCP/IP enviá-los a um dispositivo normalmente denominado *gateway*, que está ligado ao meio físico

Ethernet. A *gateway* por sua vez está ligado aos *slaves* através de redes baseadas em RS-485 ou RS-232, comunicando sob a versão Modbus RTU ou ASCII.

Além destas abordagens que são as mais utilizadas na industria, existem fabricantes de certos equipamentos que introduzem entre o *master* e os *slaves* um dispositivo adicional, de maneira a facilitar a sua implementação em aplicações ou programas. Este dispositivo adicional é também chamado *gateway*, mas não tem apenas a função de reencaminhar as mensagens para a rede, mas sim fornecer uma API para enviar as funções aos *slaves* de uma forma mais simples. Por exemplo, pode haver um conjunto de pedidos que tenham de ser enviados consecutivamente a um *slave* para executar uma certa tarefa do equipamento. Nesse caso, a API define então regras para uma função, num determinado registo e com um determinado *input*, que representa esse conjunto de pedidos a ser enviados ao dispositivo *slave*. Assim, o *master* apenas terá de efetuar um pedido com essa função, e a *gateway* ao interpretar o pedido recebido, efetua o conjunto de pedidos ao *slave*, de uma forma transparente ao *master* e à aplicação. Neste contexto, a *gateway* e os *slaves* podem até estar ligados por outros tipo de redes e comunicar sob outro protocolo, mas servem uma API baseada no protocolo Modbus. Como não foi encontrado nenhum termo técnico que represente este tipo de esquema, este irá ser referido como Modbus API neste relatório.

2.4 Redes de Sensores e Atuadores Sem Fios

Uma rede de sensores inclui três requistos essenciais [Mul13]:

- **Recolha de dados (físicos)** — através de sensores;
- **Comunicação** — entre os vários componentes da rede;
- **Capacidade de computação** — através de hardware, software e algoritmos.

Para desempenhar as funções necessárias a cumprir a estes requisitos, uma rede de sensores inclui vários componentes chamados nodos sensor ou *motes*, que já foram abordados na Secção 2.1. Além dos sensores e *motes*, uma RSSF contém um elemento com um elevado poder de computação, normalmente chamado de nodo de ligação, *sink node*, ou *gateway*. As *gateways* tem o objetivo de receber os dados encaminhados pelos *motes* e efetuar algum tipo de processamento. Este processamento pode ser composto por várias etapas, mas na maioria dos casos é efetuado o envio dos dados a serviços externos através da internet para que possam ser armazenados em bases de dados, e possam ser utilizados por outros sistemas ou plataformas.

As RSASF incluem um novo elemento na rede, o atuador que já foi abordado na Secção 2.3.2. Os atuadores recebem as ordens de execução a partir da *gateway* da rede. A comunicação entre atuadores e a *gateway* pode ser estabelecida através de tecnologias de comunicação sem fios ou por redes físicas, como o exemplo do protocolo Modbus apresentado na Secção 2.3. Em redes mais sofisticadas, os atuadores podem também ter a cacapacidade de encaminhamento, e caso a tecnologia de comunicação sem fios seja a mesma que os *motes* utilizam, estes podem comunicar entre si optimizando o processo de encaminhamento de dados.

O grande objetivo das RSASF é dotar a *gateway* ou o servidor com o qual comunica, com sistemas de apoio à decisão para que os atuadores reajam de forma inteligente consoante os dados recolhidos dos sensores. Na Figura 2.3 está representada a arquitetura de uma RSASF.

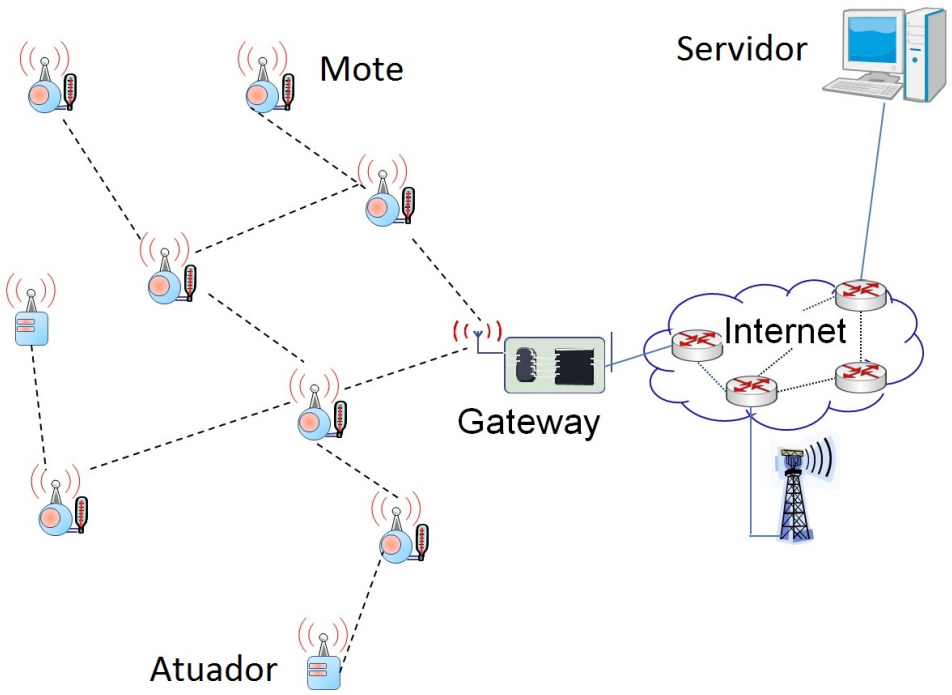

Figura 2.3: Arquitetura das RSASF

2.5 PlugSense Framework

A PlugSense Framework (PSF) é uma plataforma que permite criar, gerir e estender aplicações baseadas em RSSF. Segundo o website oficial da PSF[1], esta plataforma fornece uma estrutura capaz ultrapassar os seguintes problemas comuns no desenvolvimento de aplicações baseadas em RSSF:

- Grande dependência de capacidades muito específicas de programação.
- Longo periodo de desenvolvimento.
- Falta de *standards* de tecnologias de comunicação sem fios, existindo uma grande variedade, mas operando de maneiras diferentes.
- Dificuldade de implementar sensores de diferentes fabricantes.

[1] http://www.freedomgrow.pt/fg/pt-pt/solucoes/developmentframework.aspx

A PSF impõe-se então como um solução tecnologicamente inovadora e uma valiosa ferramenta para profissionais, que permite a criação rápida de aplicações integrando qualquer tecnologia de comunicação sem fios e qualquer tipo de sensor, independentemente do fabricante. Na Figura 2.4 encontra-se a arquitetura das aplicações criadas através da PSF.

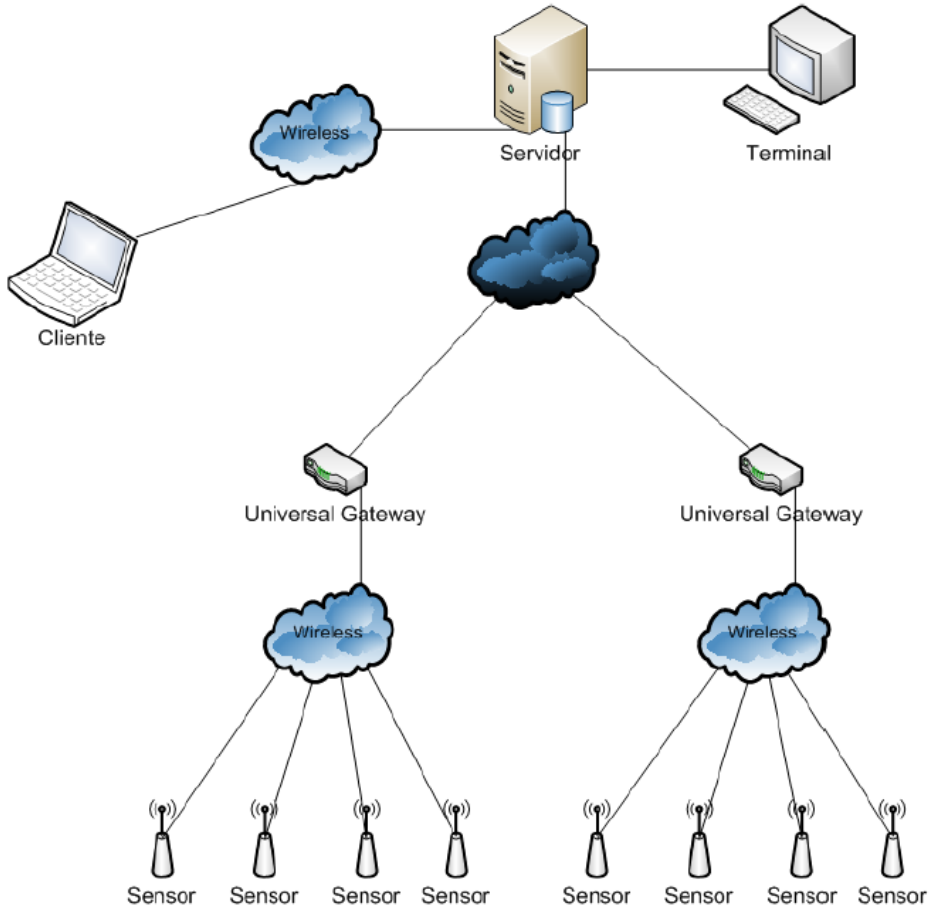

Figura 2.4: Arquitetura das aplicações geradas pela PSF

De seguida são apresentadas todas as tecnologias utilizadas pela PSF:

- Microsoft Visual Studio
- C#
- ASP.NET MVC3
- Microsoft IIS
- Mircosoft SQL Server
- HTML, CSS e JavaScript
- XML

Tendo em conta estas tecnologias e o ambiente de desenvolvimento Visual Studio, a PSF é constituída por 3 soluções que serão de seguida apresentadas. Na Figura 2.5 são representadas as soluções que foram modificadas no desenvolvimento da solução proposta, bem como algumas referências entre soluções, sendo diferenciados a amarelo novos projetos que foram criados no âmbito desta dissertação.

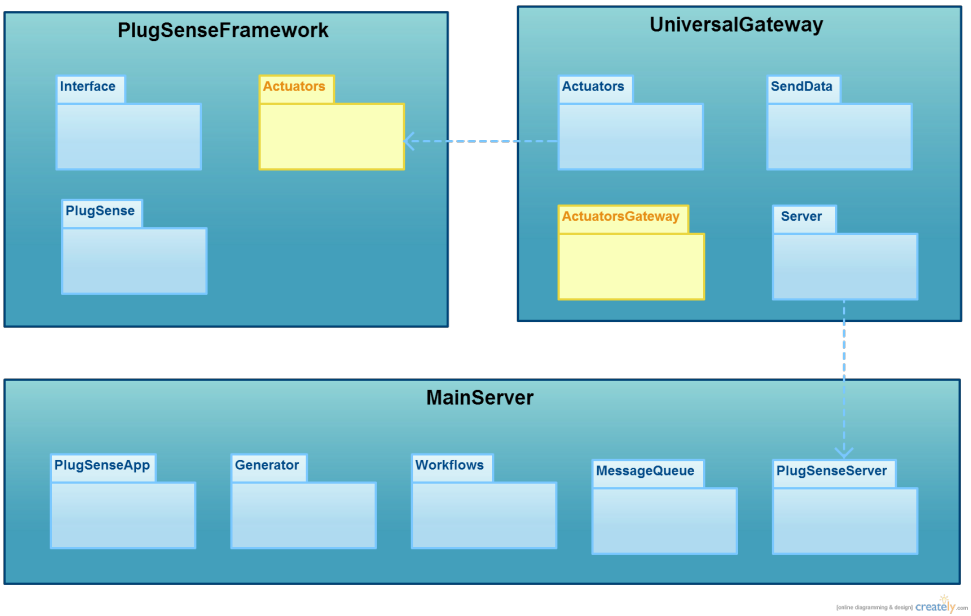

Figura 2.5: Soluções Visual Studio da PSF

- **PlugSenseFramework**

Esta solução contém a aplicação C# que permite a criação e gestão de aplicações, bem como a gestão e integração de sensores que as aplicações suportam. Para evitar mal entendidos do mesmo vocábulo para conceitos diferentes, as aplicações criadas a partir da PSF irão ser referidas como "aplicações finais" ou simplesmente "aplicação", enquanto a aplicação como ferramenta para criar aplicações finais será referida simplesmente por "Framework". O termo PSF irá ser usado para referir a Framework enquanto produto ou sistema, ou seja, o conjunto destas 3 soluções.

Para criar uma nova aplicação final, terá de ser criado um novo "Projeto " na Framework, começando por dar a escolher uma pasta reservada, onde irá ser criado o ficheiro de configuração da aplicação final, bem como os ficheiros que a constituem. Em cada projeto é possível determinar os sensores que a aplicação vai utilizar, gerir utilizadores, configurar o servidor onde a aplicação vai ser alojada, escolher estilos de interface, entre outros.

Após configurar os projetos, é possível compilá-los de maneira a que a aplicação final seja posta em execução no IIS da máquina local, existindo bastante opções de compilação como a configuração do servidor, compilação da base de dados, restauro do projeto, entre outros. Este esquema dos projetos permite então a gestão das aplicações finais, sendo possível acrescentar ou alterar módulos, mantendo o seu funcionamento global.

Além da gestão dos projetos, a Framework permite a integração de sensores para que possam ser utilizadas nas aplicações finais. Através da sua interface, é possível determinar quais os elementos que o sensor recolhe (séries) e a sua unidade de medida, os valores limite para a deteção de alertas, entre outros. É também possível gerar o *workflow* para o tipo de sensor criado. O *workflow* é uma classe que inclui uma sequência de métodos que irão ser executados no servidor, e que têm o objetivo de processar os dados recolhidos pelos sensores que são enviados em ficheiros no formato PlugSenseML pela aplicação Universal Gateway (UG). O formato PlugSenseML é um padrão para ficheiros XML que definem a estrutura dos dados a ser enviados pela UG e a ser interpretados pelo servidor.

É na UG que estão presentes as classes que irão gerar a biblioteca para a comunicação com os sensores. Uma vez gerada a biblioteca, esta tem de ser associada pela Framework ao tipo de sensor correspondente para que a UG a reconheça e consiga assim comunicar com os dispositivos e posteriormente enviar os dados ao servidor.

Ao criar um novo sensor, as suas configurações são guardados em ficheiros XML para que possam ser utilizadas nas restantes soluções da PSF. Até antes do início do desenvolvimento desta dissertação, a PSF suportava a utilização de 22 sensores.

- **MainServer**

A solução MainServer contém uma série de projetos que em conjunto formam as várias componentes das aplicações finais geradas pela Framework.

Um dos projetos desta solução com mais relevância é o PlugSenseApp, que contém a aplicação ASP.NET MVC que uma vez compilada na Framework irá ser executada no servidor e fornecerá o *website* de interação com o utilizador final. O website oferece várias funcionalidades como a gestão de entidades, associação de sensores a entidades, visualização em tempo real dos dados recolhidos pelos sensores, entre outros. Esta aplicação é então a base das aplicações finais, e é por isso chamada de "aplicação genérica", no entanto irá ser referida ao longo deste relatório simplesmente como "aplicação". A partir da aplicação é possível alterar a sua estrutura, podendo ser adicionados novos modelos e interfaces, de forma a poder criar aplicações com funcionalidades específicas, consoante o contexto em que irá ser utilizada. É através dos modelos deste projeto que a base de dados é criada, seguindo o padrão ActiveRecord.

Além do PlugSenseApp, a solução MainServer contém ainda o projeto PlugSenseServer que é de extrema relevância no sistema pois contém o webservice do tipo SOAP que irá fornecer alguns serviços essenciais à aplicação, bem como fazer a ponte entre a aplicação e a Universal Gateway. O webservice é responsável por receber os ficheiros PlugSenseML e executar o *workflow* correspondente ao tipo de sensor indicado no ficheiro.

O projeto Workflows contém as classes correspondentes aos *workflows* dos sensores. Estes podem ser criados manualmente ou gerados pela Framework. Genericamente, cada *workflow* possui dois métodos: no primeiro os atributos do ficheiro são transformados em objetos, e se houver algum erro nos dados recebidos a execução da sequência é terminada. Uma vez processados os dados, é executado o segundo método que tem o objetivo de guardar a leitura na base de dados, e

posteriormente fazer a verificação dos valores de leitura para que os alertas possam criados caso ultrapassem os limites definidos.

O projeto Generator contém a aplicação que permite criar e gerir a base de dados dos projetos finais. Através de uma série de opções, é possível acrescentar, editar ou remover tabelas de bases de dados de projetos já em execução.

- **UniversalGateway**

A solução UniversalGateway contém a aplicação C# que irá comunicar com os sensores ou *motes*, podendo esta comunicação ser estabelecida por tecnologias sem fios como Bluetooth ou ZigBee, ou mesmo por ligações físicas através de cabos—série ou USB. Esta aplicação tem como nome "Universal Gateway" (UG)

Esta aplicação tem a característica de identificar a conexão dos sensores e automaticamente transferir os dados recolhidos ao servidor, permitindo que sejam consultados em tempo real no *website*. Esta funcionalidade é uma das grandes vantagens da utilização da PSF para criar aplicações baseades em RSSF, pois uma vez integrados os sensores, basta adicionar o sensor a um projeto e compilá-lo com a opção de compilação da UG selecionada, e na próxima execução da UG deverão apenas ligar o dispositivo à maquina começando automaticamente o envio de leituras.

Para cada sensor deverá existir um projeto com o mesmo nome nesta solução. Este projeto contém a classe cujos métodos permitem a comunicação com os dispositivos. Uma vez que cada tipo de sensor contém o seu formato de dados, o código destes métodos deve ser desenvolvido manualmente, e não são gerados pela Framework. No entanto, existe um *template* para esta classe com o esqueleto dos métodos que devem ser desenvolvidos. Ao compilar o projeto, é gerado o ficheiro DLL que deverá posteriormente ser associado ao tipo do sensor na Framework. Esta associação fica registada no ficheiro XML de configuração da UG, que na próxima execução irá utilizar os métodos das bibliotecas dos sensores existentes e detetar de que tipo de sensor se trata, e comunicar adequadamente com o dispositivo.

2.6 Modelos de prevenção de doenças

Nesta secção serão apresentados os modelos de prevenção de doenças que foram introduzidos na Framework e podem ser utilizados nas aplicações finais que permite criar. Estes modelos foram aconselhados pela APAS e são doenças frequentes em culturas de peras e maçãs.

- Estenfiliose (*Stemphylium versicarium*) — Modelo BSP–Cast [eTHN]

BSP–Cast é um modelo empírico que determina o risco de infeção da Estenfiliose na pereira quando as condições ambientais se tornam favoráveis para a infeção e desenvolvimento da doença. O modelo calcula um valor de gravidade da doença de acordo com a equação (2.1), que tem como dados de *input* a duração do tempo de folha molhada (W - horas) e a temperatura média desse período (T - ^{o}C).

$$log10(S) = f(T,W) \tag{2.1}$$

onde

$$f(T,W) = -1.70962 + (0.0289 \times T) + (0.04943 \times W) + (0.00868 \times T \times W) - (0.002362 \times W^2) - (0.000238 \times T^2 \times W) - (0.002362 \times W^2) - (0.000238 \times T^2 \times W) \tag{2.2}$$

Este valor é divido pelo valor máximo de gravidade obtido para os valores de humectação e temperatura ótimos (igual a 3.7942) calculando deste modo um índice de risco (IR) que varia entre 0 e 1.

$$IR = \frac{S}{3,7942} \tag{2.3}$$

O índice de risco acumulado (IRA) é calculado totalizando o índice IR dos últimos 3 dias. O IRA é utilizado como indicação para intervenção com tratamentos fungicidas aconselhados. O parâmetro IR e IRA são calculados diariamente. O momento oportuno para se iniciar a observação recai no período de Abril a Maio, quando a planta se torna suscetível à doença.

Este modelo é uma ferramenta útil, não tanto para determinar o período mais oportuno para fazer eventuais tratamentos, mas sim para evidenciar os períodos de baixo risco e quando aumentar o intervalo entre tratamentos.

O limite a partir do qual se justifica uma intervenção fitossanitária tem o valor de 0,4. Este limite tem permitido minorar o número de tratamentos, mantendo constante e inalterada o nível de estragos na produção face à luta química tradicional (tratamentos semanais).

- Pedrado (*Venturia pirina*) — Modelo Spotts & Cervantes [eTHN]

Os dados atuais de Spotts e Cervantes, de experiências em atmosferas controladas com sementes de pereira, demonstram os efeitos da temperatura e da duração do período de humectação na infeções conidiais em sementes de pereira, folhas e frutos.

O Modelo de Spotts & Cervantes prevê as infeções por conídios de Venturia pirina em folhas como o número mínimo de horas de humectação necessárias para a infeção (W - h), com base na regressão das horas de humectação e a temperatura durante esse período húmido (T - ^{o}C):

$$W = 66.82 - (8.72 \times T) + (0.44 \times T^2) - (0.0076 \times T^3) \tag{2.4}$$

Os autores não avaliaram as condições de infeções com ascósporos, mas sugerem que devam ser similares às infeções conidiais, pelo que o modelo pode ser usado para predizer as infeções primárias por ascósporo. No entanto, a APAS desenvolveu uma adaptação deste modelo para infeções com ascósporos:

$$W = 46.77 - (6.3656 \times T) + (0.3269 \times T^2) - (0.0055 \times T^3) \tag{2.5}$$

- Piolho de S. José (*Quadraspidiotus perniciosus*) — Graus dia

Com base no somatório das temperaturas médias diárias superiores ao zero de desenvolvimento da praga (7,3°C) a contar desde 1 de Janeiro de cada ano, determina-se a emergência das ninfas da primeira geração.

A emergência das ninfas da primeira geração ocorre quando se regista 500-525°C e para a segunda geração é de 1270-1295°C.

- Cecidómia (*Dasineura pyri*) — Graus dia [GL92]

Este é uma espécie de insetos que passa por três fases: ovos, larva e adulto. As fêmeas depositam por broto, entre 3 a 30 ovos, podendo colocar entre 100 e 120 ovos no total. A fase de gestação dura entre 3 e 12 dias, consoante as temperaturas. As larvas recém-nascidas permanecem dentro da fruta, onde encontram abrigo e alimento.

O início do ataque da primeira geração acontece quando a acumulação de temperatura média desde 1 de Janeiro ultrapassa os 830°C. A segunda geração surge em finais de Abril e início de Maio de uma maneira definida, e a terceira no início de Junho. Entre Julho e Agosto, dependendo das temperaturas, as larvas formam casulo debaixo do solo perto de árvores, onde hibernam até à primavera seguinte.

- Mosca da fruta (*Ceratitis capitata*) — Modelo Bodenheimer [GL92]

Como a Cecidómia, a mosca da fruta também apresenta 3 fases da vida. Cada fêmea pica vários frutos, depositando em média 300 ovos, tendo a incubação dos ovos a duração de 2 a 4 dias. As larvas, ao nascerem, inicia rapidamente a alimentação, destruindo completamente o fruto devido ao ataque secundário de bactérias e fungos.

Em 1951, Bodenheimer conclui que a ovulação das fêmeas é determinada em função da temperatura e humidade ambiente, estabelecendo quatro "zonas" segundo as exigências climáticas da espécie e viabilidade da sua reprodução em cada uma estas zonas. Cada zona traduz-se numa escala de probabilidade qualitativa de uma geração reproduzir. A definição das zonas em função da relação entre humidade e temperatura são apresentadas na tabela 2.4.

Tabela 2.4: Modelo Bodenheimer

Zona	Temperatura (°C)	Humidade (%)
Zona ótima (A)	16-32	75-85
Zona favorável (B)	10-35	60-90
Zona não favorável (D)	2-38	40-100
Zona impossível (D)	2-40	<40

2.7 Conclusões

Após a revisão sobre RSASF e os componentes que as integram, é possível ter uma noção concreta da estrutura e características que as aplicações geradas através da Framework desenvolvida deverão apresentar.

A descrição detalhada da PlugSense Framework permitirá uma melhor compreenção do trabalho que foi efetuado nesta dissertação e que será descrito no Capítulo 4, bem como os modelos de doenças apresentados, uma vez que fazem parte de uma funcionalidade que as aplicações geradas pela Framework contêm.

Capítulo 3

Aplicações para AP

Na sequência da revisão tecnológica, neste capítulo serão apresentadas soluções baseadas em RSSF e RSASF dirigidas à AP. Embora haja vários estudos e implementações deste carácter, foi feita uma seleção através de um critério que priveligiou:

- Estudos mais recentes
- Utilização de atuadores
- Diferentes áreas da agricultura

Após a apresentação e uma breve apreciação destas aplicações, será feita uma análise crítica e comparativa em termos da flexibidade e extensibilidade. Com base nesta análise, será também refinado o objetivo do desenvolvimento da Framework, bem como apresentados alguns desafios que se impuseram.

3.1 RSSF para horticultura de precisão no Sul de Espanha

Em [RSS+09] é descrita uma experiência conduzida para a introdução de uma RSSF num terreno de horticultura de 1000ha localizado no Campo de Cartagena na Região da Murcia no Sudoeste de Espanha, contendo 250 campos de colheitas espalhados no terreno com alguns quilómetros de distância entre eles. O objetivo desta implementação era fornecer a este terreno uma infraestrutura que determinasse em tempo real, e por colheita, as condições e necessidades de água das colheitas, para que pudessem ser feitas as decisões apropriadas.

Foram definidas duas redes de sensores. A primeira recolhe a temperatura, viscodidade e salinidade do solo, e a segunda recolhe temperatura e humidade do ar. Adicionalmente, um sensor sem-fios foi colocado isoladamente num reservatório para medir a salinidade da água que é utilizada na rega das colheitas. Estas subredes e o sensor isolado enviam dados pelo nodo *gateway*

apropriado para uma estação-base localizada nos gabinetes centrais do campo, onde as decisões estratégicas sobre as colheitas eram tomadas.

Foi também desenvolvida uma aplicação para controlar todos os dispositivos e registar toda a informação recebida numa base de dados relacional, que era utilizada para tomar decisões sobre a irrigação. O sistema desenvolvido foi testado em duas fases, a primeira em ambiente de laboratório, e a segunda no terreno. Na primeira fase, as duas sub-redes foram implementadads apenas com quatro sensores nos nodos ambiente e de solo, em conjunto com os nodos gateway, estação base e repetidores. A principal função desta fase era validar o *software* e *hardware* propostos. A segunda fase, conduzida nas condições reais do campo, teve como objetivo testar o desempenho funcinal dos dispositivos desenvolvidos, como alcance, robustez e flexibilidade.

A arquitetura do sistema desenvolvido foi desenhada de maneira a que possa ser implementado em vários campos localizados dispersamente (até 10km de distância da base central), tornando-o bastante complexo para aplicar em terrenos com uma topologia mais simples e de menores dimensões. Apesar dos componentes utilizados no *hardware* (*motes*) desenvolvido terem sido especificados, apenas foram testados com ligação a 3 tipos de sensores, não fornecendo uma especificação dos módulos que permitiriam incluir novos sensores no sistema, tornado-o assim inflexível.

3.2 Controlo de Irrigação de Plantações utilizando uma RSASF

Em [aSYN+10], é apresentado um controlo de custo eficiente para a irrigação de plantações baseado em RSASF. A eficiência de custo atingida pelo sistema é conseguida através de uma arquitetura pioneira de sensores sem fios e atuadores, e os seus protocolos para desenvolvimento das redes. No desenvolvimento do sistema foi utilizada uma abordagem modular, dividindo o problema em partes mais pequenas:

- **Nodo Sensor (*Sensor Mote/Node*)** — Tem como objetivo medir propriedades físicas através de sensores bem como de interfaces de ligação com sensores exteriores.
- **Nodos Atuador (*Actuator Node*)** — Tem como objetivo ligar e desligar um dispositivo atuador.
- **Nodo Ligação (*Sink Node*)** — Este nodo tem como objetivos receber pacotes de dados dos nodos sensores via Zigbee e transferi-los para um computador através de uma interface de série, e enviar pacotes de pedidos aos nodos atuadores gerados pelo sistema de suporte à decisão através da RSASF.

Para permitir estas comunicações, foi desenvolvido o protocolo "ProtoSense", baseado num protocolo broadcast. Este protocolo é considerado eficiente em gastos de energia, uma vez que reduz comunicações extra e introduz o padrão *request-acknowledgment*.

No sistema de suporte à decisão, são definidos diferentes níveis máximos de temperatura, humidade do solo, humidade do ar e luminosidade. As decisões para os atuadores de irrigação são feitas quando os níveis anteriormente referidos forem ultrapassados. A decisão de atuação

traduz-se em ligar o dispositivo de irrigação durante um período de tempo definido, que é enviado no pacote de atuação. O *software* para a irrigação permite ver a atuação no formato gráfico.

Como os autores indicam, este sistema não apresenta extensibilidade, e apesar dos conceitos se adequarem à AP, não foi testada a sua aplicação real num ambiente agrícola.

3.3 Aplicação baseada em RSSF para poupança de água na irrigação

Em [FZG07] foi desenvolvido uma RSSF que se aplicava num sistema de irrigação e poupança de água. A RSSF era constituida por duas camadas baseando-se nas necessidades e capacidades do sistema formado por irrigação por pulverização e canalização. Foi conseguida uma expressão para o número mínimo de sensores necessários para cobrir a área através da relação entre o alcance de cada sensor e o alcance dos pulverizadores.

Foi apresentado o método de transmissão de dados bem como a estrutura da rede e a distribuição dos nodos. A fiabilidade da transmissão de dados foi assegurada através de um método de diagnóstico de falhas em camadas.

A fórmula encontrada que permite calcular o número mínimo de sensores para cobrir o terreno é uma mais-valia. No entanto, não foi apresentada uma análise rigorosa do ganho em termos de poupança de água que se alcançou. A arquitetura do sistema foi também desenhada de uma maneira muito específica, sendo muito difícil adaptar esta solução noutros ambientes que tenham outras infraestruturas de fornecimento de água, pois nem todos utilizam canalização e oleodutos.

3.4 Sistema de suporte à decisão de fertilização ideal

Em [HWH+10], é descrito o desenvolvimento e implementação de um sistema de apoio à decisão de aplicação fertilizantes utilizando uma LAN de sensores sem fios, o protocolo IEEE 802.11 (WiFi) e análise de servidor por GPS (GIS). Os sensores foram utilizados para adquirir dados em tempo real de viscosidade de solo, condutividade, temperatura, valores de PH, temperatura do ar, humidade, concentração de CO2, iluminação, etc. O sistema foi concebido utilizando a estrutura *Browser/Server* para providênciar interatividade de alto nível. O GIS implementado foi utilizado para interpolar dados de parcelas experimentais mais pequenas para parcelas maiores, e assim explorar redução de dados e conservação de energia.

Foi utilizado o teste "3414" [CZ06] no sistema de decisão de fertilização, utilizando 3 fatores (nitrogénio, fósforo e potássio) e 14 tratamentos de fertilizantes. O teste de eficiência de fertilização do "3414" é utilizado predominantemente em sistemas de apoio à decisão de fertilização, sendo um modelo estatístico sobre fertilização e produtividade de colheitas. Através deste modelo, é possível calcular níveis ótimos de produtividade de colheitas e correspondente quantidade de fertilizantes [YS04].

O sistema de suporte à decisão apresenta diagramas de sequência dos algoritmos que implementa. No entanto, não apresenta de forma concreta a forma como os dados são representados, sendo portanto os algoritmos descritos de uma forma apenas conceptual. As bases de dados de

conhecimento sobre fertilizantes que mencionam também não estão devidamente referênciadas, não dando uma noção que tipo de conhecimennto contêm, bem como a sua orgiem. Apesar de referirem que este sistema foi testado numa instalação agrícula em Aksu, na província chinesa Xinjiang, não existe referência aos dispositivos que foram utilizados para recolher dados do solo, tornando o sistema inflexível.

3.5 Ambiente interior inteligente e sistema de gestão de energia para estufas

Em [KSD+10], foi desenvolvido um ambiente inteligente e um sistema de gestão de energia para estufas onde se monitorizou a luminosidade, temperatura, humidade, concentração de CO2 do interior da estufa. Foram desenvolvidos dois controladores baseados na lógica *fuzzy*, utilizando pontos de referência climáticos interiores. Foram controlados atuadores, na forma de unidades de aquecimento, janelas controladas a motor, cortinas controladas a motor, luz artificial, botijas de enriquecimento de CO2 e válvulas de vaporização de água.

Para modelar o ambiente interior da estufa foi utilizado o modelo matemático TRNSYS [SEL06], tendo sido feitas as alterações necessárias no modelo para modelar o equilíbrio térmico, equilibrio do nível de CO2 e ganhos de calor por radiação, sendo todas as equações especificadas. Como instalação e validação do sistema desenvolvido foi utilizada uma estufa localizada no Instituto Mediterrâneo de Agrononomia da Chania. A instalação da plataforma é baseada em LonWorks [Cor96], oferecendo interoperabilidade, expandibilidade e flexibilidade. Os controladores *fuzzy* foram desenvolvidos em MatLab e os testes de TRNSYS incluídos numa topologia em par entrelaçado da LonWorks (FTT-10A).

Em suma, todos os algoritmos de controlo são especificados. A utilização da plataforma LonWorks apresenta, segundo os autores, flexibilidade e extensibilidade no sistema. No entanto tal torna-se difícil sem uma especificação do *software* dos nodos intermédios, bem como os protocolos de comunicação entre os vários dispositivos do sistema, os quais não são fornecidos. Além disso, o desenvolvimento de sistemas na plataforma LonWorks é bastante complexa e requer conhecimentos técnicos muito específicos, tornando-se difícil adotar esta solução em grande escala.

3.6 Conceção e teste de nodos para aquisição de dados num terreno agrícola baseados numa RSSF

Em [CLea09] foi apresentado um sistema de nodos concebido para recolher informação de um terreno agrícola utilizando uma RSSF. Este estudou combinou nodos de ligação e nodos *gateway* com um sistema de processadores incorporados. Os nodos da rede foram distribuídos regularmente nas regiões do terreno recolhendo informação da composição do solo. Os nodos formaram a rede e a informação era transmitida para o nodos *gateway* que a apresentavam e armazenavam dinamicamente.

Foram testadas antenas com várias alturas: 0.5, 1.0, 1.5 e 2.0 m. Para estudar a distância de transmissão do sinal de rádio com diferentes períodos de crescimento, foram feitas experiências durante três períodos frequentes de crescimento de calor: semeação, junção e poda. Foi encontrada a relação entre a eficiência da distância de transmissão do sinal de rádio em diferentes períodos de crescimento de calor bem como a altura ideal da antena.

Este estudo concentrou-se apenas na monitorização do solo, excluindo outros fatores que possam ser relevantes noutras áreas da agricultura. Além disso apenas foi testada a comunicação por rádio, que apenas é relevante em terrenos grandes com distâncias significativas entre os nodos de ligação de nodos *gateway*. A duração das fases de crescimento de calor que foram testadas são específicas do tipo de colheitas que a cultura produzia, podendo diferência para outras culturas, limitando assim a abrangência deste estudo.

3.7 Conclusões/Objetivos

Na Tabela 3.1 encontra-se uma comparação entre as aplicações analisadas segundo os critérios que as diferenciam e que a Framework pretende cumprir. De seguida é feito um resumo das falhas de flexibilidade e extensibilidade estas aplicações, bem como uma definição precisa dos objetivos que se pretendem alcançar com o desenvovimento desta Framework.

Tabela 3.1: Comparação entre aplicações

Característica	3.1.1	3.1.2	3.1.3	3.1.4	3.1.5	3.1.6
Utilização de atuadores	-	✓	✓	-	✓	-
Especificação de sensores	✓	-	✓	-	✓	✓
Especificação de algoritmos de decisão	-	✓	✓	✓	✓	✓
Especificação de protocolos de comunicação	-	✓	-	✓	-	-
Hardware personalizado	✓	✓	-	-	-	-
Extênsibilidade	-	-	-	-	✓	-
Flexibilidade	-	-	-	-	✓	-

Os sistemas abordados que não incluem atuadores, reduzem a capacidade do sistema para apenas monitorização, não contendo os módulos necessários de extender as suas funcionalidades ao nível de controlo. Os que incluem atuadores nas arquiteturas, não especificam qual o *hardware* utilizado, sendo estes dispositivos referênciados apenas pela sua função genérica.

Apesar dos sistemas e aplicações apresentados nestes artigos conterem diagramas de arquitetura, sequência de funcionalidades e protocolos de comunicação, nenhum apresenta um manual de utilização e instalação. Além disso, nenhum apresenta uma especificação completa do *software* do sistema, incluindo modelação de bases de dados e lógica da camada de negócio, tornando impossível uma adopção em grande escala de cada uma destas aborgens por um agricultor comum, bem como uma verdadeira extensibilidade e flexibilidade das aplicações.

A ausência da especificação dos protocolos de comunicação entre dispositivos e módulos do sistema também representa uma falha de flexibilidade, uma vez que não deixam indicação de como implementar novos dispositivos e módulos para efetuar outras funcionalidade e enriquecer as aplicações.

Tendo em conta estes fatores, o grande objetivo da aplicação, será a capacidade de poder ser adquirida em grande escala na indústria agrícola, tendo portanto de abranger (na forma genérica) o máximo de funcionalidades que são essências na AP, nomeadamente uma grande variedade de sensores, que cubra todos os elementos físicos de medição possíveis na agricultura. Consequentemente, terá de suportar um número suficiente de nodos de ligação necessários a qualquer tipo de rede, bem como oferecer diferentes preços e consumos energéticos na sua aquisição e no ciclo de vida do sistema.

Além de oferecer um vasto leque de sensores, as aplicações deverão também incluir atuadores para garantir o controlo do ambiente agrícola. Os principais atuadores que se pretende incluir são sistemas de irrigação e de aplicação de fertilizantes, devido à sua importância essêncial em qualquer cultura agrícola.

Para poder ser adquirida e utilizada em grande escala, a aplicação (genérica) deverá ser flexível, no sentido de poderem ser feitas alterações em todos os módulos que incorpora, como alterações na interface gráfica, substituição de dispositivos, etc. Também deverá apresentar extensibilidade, no sentido de poderem ser adicionados facilmente novos componentes e módulos, e desta forma acompanhar o avanço técnológico bem como suportar novas necessidades impostas por avanços do setor agrícola. Com estas propriedades, será possível conceber facilmente e rapidamente soluções "à medida" para utilizadores em particular, a partir da aplicação genérica.

Tendo em conta a descrição da PSF feita na Secção 2.5, esta surge como proposta de solução aos problemas de flexibilidade e extensibilidade acabados de referir. No entanto, a PSF apenas contém os módulos que permitem gerir sensores, sendo por isso necessário desenvolver e integrar novos módulos e funcionalidades na PSF, bem como extender os que já detem, de maneira a que esta possa suportar o desenvolvimento de sistemas e aplicações que incorporam atuadores, e os permitam gerir de acordo com os dados recolhidos pelos sensores. Além disso, algumas estruturas e funcionalidades da aplicação genérica terão de ser adaptados, uma vez que esta possui alguns padrões que limitam a utilizada da aplicação no contexto agrícola. Entre outros, um exemplo de uma característica da aplicação que não se adequa a um cenário agrícola é o facto de apenas poder associar um sensor a uma entidade. Tendo em conta que as entidades das aplicações serão as parcelas de terreno onde as colheitas estão plantadas, haverá a possibilidade de ser necessário associar um sensor a mais que uma parcela de terreno. Apesar de até ao início do trabalho desenvolvido nesta dissertação a PSF permitir a utilização de 22 sensores, existem dados ambientes que são de extrema utilidade para a atividade agrícola que não são recolhidos pelos sensores disponíveis. Desta forma, será necessário integrar novos sensores que alarguem a capacidade geral de monitorização de dados físicos e ambientais das aplicações geradas a partir da Framework.

Capítulo 4

Desenvolvimento da solução

Neste capítulo será apresentada a especificação e o processo de desenvolvimento da Framework. Tendo em conta a motivação e o conceito apresentados no Capítulo 1 e os objetivos apresentados na Secção 3.7, o processo de conceção envolveu diferentes fases com objetivos de desenvolvimento específicos que serão abordados separadamente nas secções deste capítulo.

Sintetizando os objetivos do desenvolvimento, numa primeira fase foram feitas alterações na aplicação genérica da PSF para que esta apresente estruturas e funcionalidades dirigidas à AP, bem como a integração de sensores que recolhem dados que tenham impacto no processo de crescimento de colheitas. Acrescentaram-se mecanismos e estruturas aos vários módulos da Framework para o suporte de atuadores.

Na Figura 4.1 é apresentada a nova arquitetura das aplicações geradas pela Framework que foi conseguida através do trabalho efetuado no âmbito desta disertação, tendo em conta a arquitetura original apresentada na Secção 2.5.

4.1 Especificação e requisitos da aplicação

Nesta secção serão abordadas as funcionalidades acrescentadas à aplicação genérica da PSF, bem como alterações que foram feitas de modo a que a aplicação fosse adequada ao contexto da AP. Na Figura 4.2 encontra-se um modelo de domínio onde estão representadas as entidades da aplicação genérica criada pela Framework e as suas relações.

A aplicação suporta dois tipos de utilizador: utilizador regular e administrador. O administrador terá acesso a todas as entidades do sistema e poderá também executar todo o tipo de funcionalidades sobre elas, bem como configurar definições globais. O utilizador regular não tem acesso às configurações do sistema, e apenas pode aceder às funcionalidades que dizem respeito à sua conta e perfil, bem como às operações sobre os elementos dos grupos a que pertence, consoante as suas permissões.

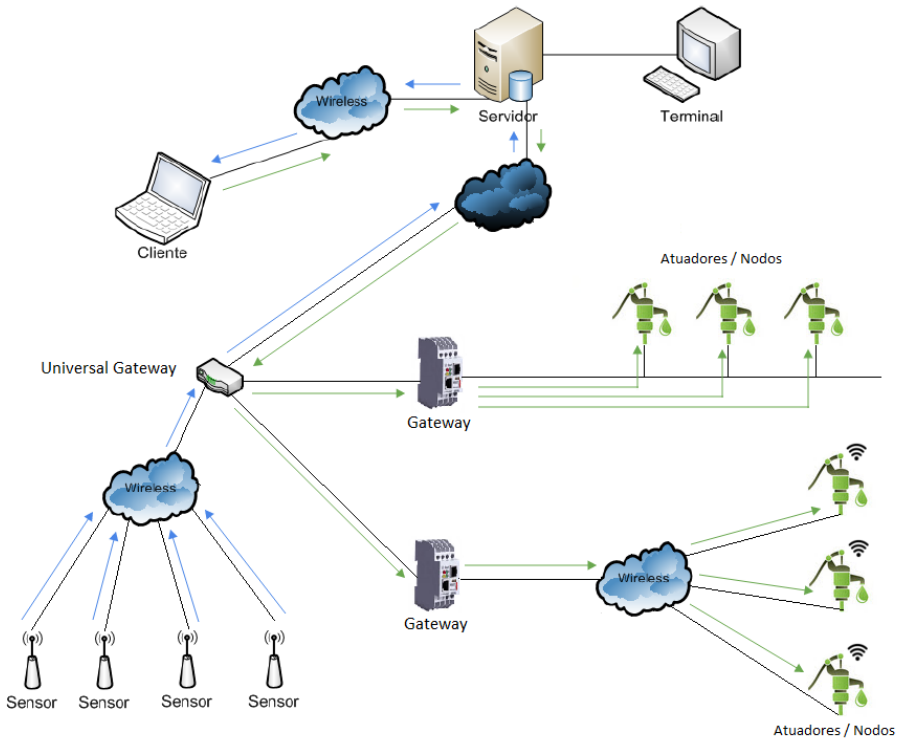

Figura 4.1: Nova arquitetura das aplicações geradas pela Framework

Os grupos apresentam-se como uma maneira de associar utilizadores e parcelas de terreno, existindo 3 tipos de permissões que os utilizadores podem ter num grupo, e que definem que tipo de operações podem executar nas parcelas e outros utilizadores. As funcionalidades do sistema e o seu acesso consoante os 3 tipos de permissões são apresentadas na tabela 4.1, tendo em conta que um utilizador sem permissões apenas pode visualizar a informação sobre as parcelas do grupo. Os utilizadores podem ser membros de vários grupos, enquanto uma parcela está associada a um único grupo.

O conceito de subgrupo foi introduzido na aplicação no sentido de oferecer uma hierarquização mais robusta de utilizadores e permissões nos grupos do sistema, e uma maior organização das parcelas. Desta forma, os utilizadores de um determinado grupo têm acesso a todas as parcelas, utilizadores e funcionalidades dos subgrupos, com o mesmo nível de permissão. Já os utilizadores dos subgrupos não têm visibilidade sobre as entidades do supergrupo dos grupos a que pertençam. Os subgrupos também têm um papel vital na associação de sensores e atuadores às parcelas de terreno, como veremos de seguida. Assim, também é possível permitir utilizadores regulares gerir os seus próprios grupos, uma vez que só um administrador pode criar um grupo que não seja subgrupo de outro.

Na versão original da aplicação genérica, cada sensor apenas podia estar associado a uma entidade (parcela) ou a um utilizador. Esta regra não vai de encontro às necessidades de um ambiente agrícola, pois um sensor pode estar associado a várias parcelas, e a associação de sensores

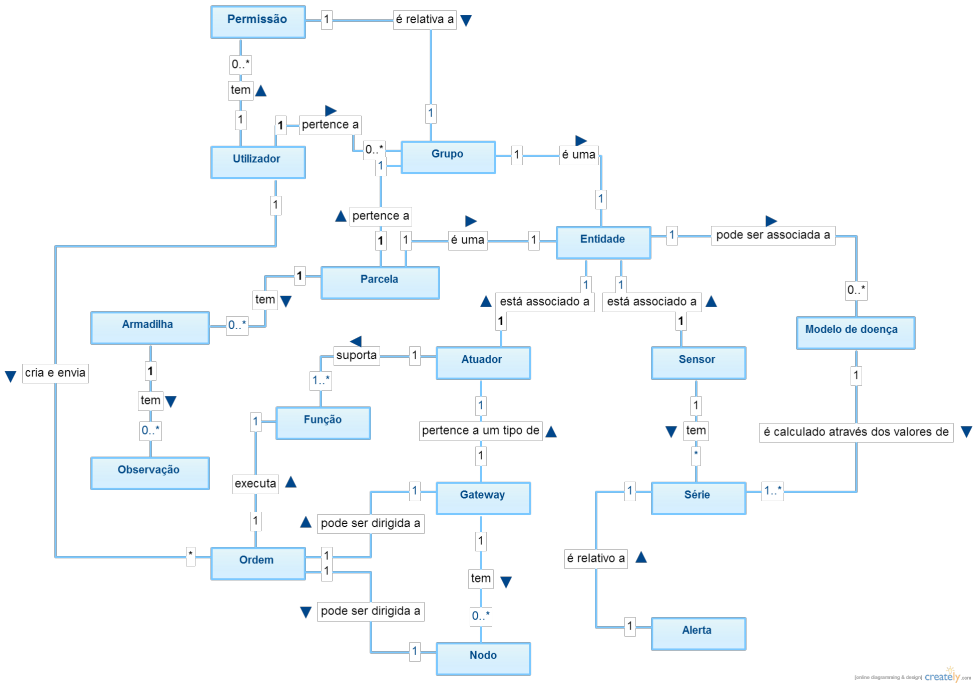

Figura 4.2: Modelo conceptual da aplicação

a utilizadores apenas faz sentido noutros contextos, como monitorização de sinais vitais. Desta forma, adaptou-se a lógica da associação de sensores para que além de poderem estar associados a uma parcela, possam estar associados a um grupo. Por exemplo, se um utilizador desejar associar dois sensores a dois conjuntos de parcelas diferentes, pode criar dois subgrupos onde irá criar os dois conjuntos de parcelas, e posteriormente associar o sensor a cada subgrupo. Uma vez que o utilizador é membro do supergrupo, terá acesso a todas as parcelas dos subgrupos.

Por cada série de dados de um sensor são determinados por um administrador valores mínimos e máximos, que uma vez ultrapassados geram alertas que são demonstrados aos utilizadores que têm acesso às parcelas ou grupos a que o sensor está associado. Os alertas podem ser de aviso ou de perigo, podendo estes parâmetros ser editados por um administrador. Depois de serem apresentados, os alertas podem ser arquivados de seguida, à exceção de um alerta de perigo, que só pode ser arquivado quando os valores passarem a níveis correspondentes a um alerta Médio.

Foi introduzida na aplicação a funcionalidade de definir e observar a localização geográfica de grupos e parcelas. Através de uma barra de pesquisa é possível navegar facilmente no mapa, oferecendo aos utilizadores a capacidade de definir e editar as áreas dos grupos e parcelas a que o utilizador tem acesso. Os mapas têm como base o OpenStreetMap, com a opção de utilização da camada de visualização satélite do Google Maps.

Foi criada também a opção de gerar anotações genéricas em cada parcela, bem como a manutenção de armadilhas de insetos. As armadilhas de insetos são utilizadas com muita frequência em determinadas culturas agrícolas. Estas são colocadas nas parcelas, e periodicamente são feitas observações sobre o número de insetos capturados, de forma a obter uma noção do número de

insetos médios que circulam nas parcelas, podendo prejudicar algumas plantas. Com a funcionalidade introduzida na aplicação, é possível criar armadilhas e adicionar observações, sendo possível visualizar através de um gráfico a média de ocorrências diárias em cada parcela. Através das anotações é possível manter o registo de características sobre as colheitas adequadas a cada atividade agrícola, como a quantidade em kg, o número de unidades desperdiçadas, etc.

Uma das funcionalidades mais inovadoras da aplicação, é a visualização de modelos de prevenção de doenças e pestes em tempo real. Foram implementados na aplicação os 4 modelos apresentados na Secção 2.6, podendo estes ser associados a grupos ou parcelas. Cada modelo é calculado com base num conjunto de dados ambientais, como tal, para associar um modelo a um grupo ou parcela, terão também de ter associados sensores com as séries de dados necessárias para o cálculo desse modelo. Os cálculos de cada modelo são efetuados nos *workflows* dos sensores a que as séries de dados estão associadas.

A fonte destes fatores é dada a escolher ao utilizador dentro dos sensores que tem disponíveis no grupo ou parcela. Uma vez associados, os fatores de risco podem ser consultados em tabelas e gráficos na página do grupo ou entidade, podendo consultar um histórico dos resultados destes modelos por dia.

A aplicação suporta também a gestão e interação com atuadores, sendo estas funcionalidades e a arquitetura que as suportam abordadas na Secção 4.3.

4.2 Integração de sensores

Como foi dito na Secção 3.7, um requisito fundamental para que as aplicações geradas pela Framework possam ser úteis para a AP, é a integração de um conjunto de sensores que efetuem medições de elementos relevantes no processo de crescimento das colheitas.

Foi feita uma pesquisa de sensores que pudessem ser adequados neste âmbito. No entanto, a APAS possui um conjunto de estações meteorológicas Vantage Pro21[1] fabricadas pela Davis Instruments, que estão espalhadas por algumas regiões do centro de Portugal, e que foram disponibilizadas para o desenvolvimento deste trabalho. Esta parceria facilitou assim o processo de seleção de sensores a integrar, pois as Vantage Pro2 satisfazem por completo os requisitos definidos para a aplicação.

Cada estação possui um conjunto de sensores que recolhem 44 fatores ambientais, efetuando leituras a cada 15 minutos. De forma a não tornar a enumeração demasiado extensa, serão de seguida apresentados os dados ambientais mais relevantes e agrupados em categorias.

- **Vento** — Direção, Velocidade média e máxima e Taxa de aceleração.
- **Solo** — As estações possuem quatro sensores que recolhem dados subterraneamente. Estes sensores estão colocados a quatro níveis de profundidade diferentes, e são sensores de temperatura e tensão do solo.

[1] http://davisnet.com/weather/products/weather_product.asp?pnum=06152

- **Humidade** — Humidade ambiente e temperatura para formação de orvalho.
- **Precipitação** — Total e taxa de precipitação.
- **Temperatura** — Máxima, média e mínima.
- **Pressão Barométrica**
- **Densidade do ar**
- **Folha** — Dois sensores que simulam a folha de uma planta, indicando a quantidade de água na superfície. Os dois sensores são colocados a altitudes diferentes.
- **Solar** — Radiação solar, energia solar e índice UV.

A cada 15 minutos, as leituras são enviadas a um dispositivo denominado DataLogger (DL). A comunicação entre a estação e o DL é feita através de RF, sendo a distância máxima entre ambas de 300 metros. O DL possui memória interna que permite guardar dados de aproximadamente 3 semanas, e após atingido esse limite os dados mais antigos começam a ser substituídos por dados mais recentes.

A estação possui um painel solar que alimenta todos os sensores das estações. No entanto, possui uma bateria auxiliar com autonomia para 9 meses que permite o funcionamento dos sensores durante a noite. Os DLs podem ser ligados à corrente elétrica e têm também suporte de pilhas no caso de a corrente falhar. O fornecedor oferece juntamente com este dispositivo uma caixa de estanque para que este possa ser colocado numa parcela de terreno ao ar livre, estando assim protegido contra condições climatéricas adversas. Na Figura 4.3 encontra-se um DL na caixa protetora juntamente com um modem GSM, uma estação Vantage Pro2 e os sensores do solo. Todas estas fotos foram tiradas nas instalações da APAS.

Figura 4.3: DataLogger e Vantage Pro2

O DL tem também a função de transferir todos os dados guardados na sua memória para computadores ou outros dispositivos externos. As transferências são feitas através do software WeatherLink (WL) que a própria Davis disponibiliza, sendo o protocolo de transferência de dados desconhecida e tratada exclusivamente por este software.

A única garantia para que comuniquem é que seja possível estabelecer uma ligação ponto-a-ponto entre o DL e o computador remoto onde WL está a ser executado. Existem 3 formas possíveis de estabelecer esta ligação:

- **Cabo série ou USB** — Requer a ligação física entre o computador e o DL.
- **TCP/IP** — Requer que o DL tenha acesso à Internet.
- **Modem GSM** — Requer que o DL tenha acesso à rede telefónica ou a utilize um cartão SIM.

Antes de poder configurar a comunicação, é necessário criar a estação no WL, escolhendo um nome para a estação e um caminho, sendo criada uma nova pasta com o mesmo nome no caminho escolhido. Nesta nova pasta irão estar os ficheiros de configuração da estação, a base de dados local que guardará os dados recebidos pelo DL, entre outros.

No início do processo de integração das estações na Framework, os responsáveis da FreedomGrow contactaram a Davis Instruments no sentido de determinar a possibilidade de fornecerem o protocolo de comunicação entre o WL e os DLs. Deste forma, a comunicação poderia ser feita diretamente entre a UG e os DLs para uma maior integração das estações na Framework e menor dependência de elementos externos. No entanto, e por razões que não foram especificadas, o protocolo não foi fornecido, emergindo a necessidade de encontrar uma alternativa para adquirir os dados das estações e enviá-los ao servidor de forma automática, para que possam ser observados no website em tempo real, à semelhança dos restantes sensores da Framework.

O WL permite que na configuração das estações, sejam definidos períodos nos quais as leituras dos últimos dois dias registadas nos DLs sejam transferidos automaticamente para a base de dados local. Além desta funcionalidade, existe uma opção que permite após a a transferência de dados permite exportar automáticamente a informação dos últimos dois dias presentes na base de dados local para um ficheiro "download.txt" que é criado na pasta base da estação. Tirando partido destas opções de configuração (apresentados na Figura 4.4) foi possível criar mecanismos que permitem a transferência automática de informação entre o WL e o servidor da aplicação.

Para isso foi acrescentado um módulo à UG que permite lêr, interpretar e por fim enviar os dados presentes nos ficheiros de texto exportados pelo WL. Foi acrescentado ao webservice um método que permite adquirir a informação das estações existentes na aplicação. Assim, ao inicar uma sessão do webservice no arranque da UG, a informação das estações é descarregada e apresentada ao utilizador. Posteriormente, o utilizador pode associar a cada estação o ficheiro "download.txt" correspondente que está presente na pasta da estação definida no WL. Esta informação fica guardada num ficheiro XML para que as configurações das estações estejam sempre disponíveis ao iniciar a aplicação.

Após a atribuição dos ficheiros, a UG verifica periodicamente qual foi a última modificação do ficheiro (significando que o WL exportou novos dados), e em caso afirmativo processa o ficheiro para que a informação seja enviada para o servidor, mantendo os dados dos sensores atualizados. A aplicação tem também a função de interpretar e enviar dados de ficheiros manualmente, o que

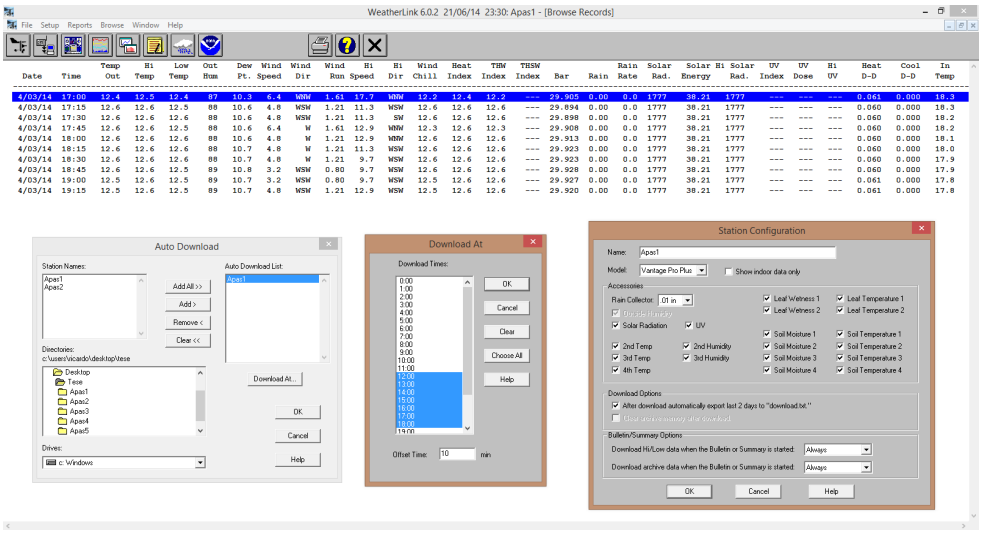

Figura 4.4: Configuração da uma estação no WL

pode ser útil em casos de falhas de comunicação das estações, ou envio de dados de períodos anteriores ao do primeiro momento de configuração da UG. Uma imagem representativa deste mecanismo pode ser observada na imagem 5.2 do Capítulo 5, embora tenha sido retirada de uma adaptação da UG genérica para a aplicaçao desenvolvida para a APAS.

O formato de envio da informação pela UG é o PlugSenseML que é utilizado nos restantes sensores, sendo executado no servidor o respetivo *workflow* que tratará de determinar erros nos dados recebidos e armazená-los na base de dados para que possam ser apresentados no website.

4.3 Suporte de atuadores

Um dos grandes objetivos deste projeto é acrescentar à PSF a capacidade gerir e integrar atuadores de forma flexível, à semelhança dos mecanismos que utiliza com os vários sensores que suporta.

Após uma análise de atuadores existentes no mercado, verificou-se que existe bastante variedade de protocolos de comunicação e funcionamento geral de atuadores que costumam ser utilizados na AP, mas também se observou que existe uma enorme oferta em dispositivos que funcionam sobre o protocolo Modbus. Este protocolo tem a vantagem de tanto poder ser implementado por tecnologias de comunicação sem fios como em ligações físicas entre as máquinas, oferecendo assim flexibilidade a nível das necessidades do sistema onde se irá utilizar a aplicação.

Foram efetuados vários contactos com fornecedores de equipamentos de rega com o intuito de adquirir e integrar este tipo de atuadores na PSF utilizando os módulos desenvolvidos, tendo sido considerado como garantida a aquisição de um equipamento em específico. No entanto o processo de negociação complicou-se e prolongou-se ao ponto de a empresa considerar que não era possível adquirir o equipamento a tempo de levar a cabo a sua integração. Apesar deste percalço foi decidido continuar com o desenvolvimento do módulo para a criação e gestão dos atuadores

baseados em ModBus, principalmente porque o objetivo da Dissertação é criar uma Framework para integrar diferentes atuadores oferecendo assim a flexibilidade pretendida. Em segundo lugar, a empresa tem em sua posse um sensor que comunica através desta tecnologia, e desta forma foi possível testar os tipos de mensagens presentes no protocolo com um dispositivo real, que apesar de não ser possível visualizar as ações que executa, dá a garantia que o processo de comunicação é efetuado corretamente com as ferramentas que se criaram.

Para levar a cabo a conceção deste grande requisito, foram criados novos elementos e mecanismos nas 3 soluções da Framework. Estas novas características da Framework podem ser divididas em duas partes que irão ser abordadas separadamente para uma melhor compreensão dos conteúdos desenvolvidos.

4.3.1 Integração de atuadores na Framework

A solução PlugSense Framework contém a aplicação responsável pela gestão de sensores e que permite criar aplicações finais. Através de uma extensa análise e um processo de refinação, a estrutura de um atuador na Framework foi definida e é representada na Figura 4.5, onde as classes correspondem ao formato dos elementos do ficheiro XML onde a informação sobre os atuadores é guardada. Foi criado nesta solução um novo projeto Actuators onde estão definidas as classes que permitem representar e incluir os atuadores nas restantes soluções da Framework. Após a definição desta estrutura, foi desenvolvida toda a interface com o utilizador que permite criar e editar cada um dos elementos de um atuador. Ao criar um novo atuador através da Framework, são criados vários documentos XML com a sua informação em vários projetos das soluções da PSF, para que estas consigam interagir com este novo tipo de entidade.

Na Figura 4.6 são apresentados o separador principal dos atuadores na Framework, bem como a janela que permite integrar um novo atuador. A interface que permite configurar funções para um novo atuador é apresentada na Secção 5.3 do próximo capítulo, juntamente com um exemplo de integração de um dispositivo Modbus.

Como pode ser observado nas Figuras 4.5 e 4.6, cada atuador contém alguma informação essencial como o nome, descrição, número máximo de dispositivos (*slaves*), entre outros. A opção "Node Offset" representa o offset a aplicar no caso do atuador utilizar o esquema Modbus API referida na Secção 2.3.2. O conjunto de opções presentes nas classes Communication e Controller são definições utilizadas na configuração da porta-série da UG e no controlador Modbus, respetivamente. O controlador Modbus é um objeto de uma classe externa utilizada nos métodos gerados pela Framework que permite executar com alto nível as funções Modbus numa porta-série.

Como foi dito, é possivel criar funções para cada atuador, que representam um comando ou ordem que o atuador suporte. Cada função está associada a uma das funções disponibilizadas pelo protocolo Modbus, e uma vez criado o atuador, são gerados métodos para cada função numa classe gerada e dedicada ao novo atuador. Caso a função seja de escrita, é necessário adicionar pelo menos um parâmetro de *input*. Nos parâmetros de *input* é possivel determinar o seu tamanho, nomeadamente 16, 32 ou 64 bits. A conversão de dados é feita automaticamente no código do método gerado para a função. Além dos parâmetros de input que serão automaticamente enviados

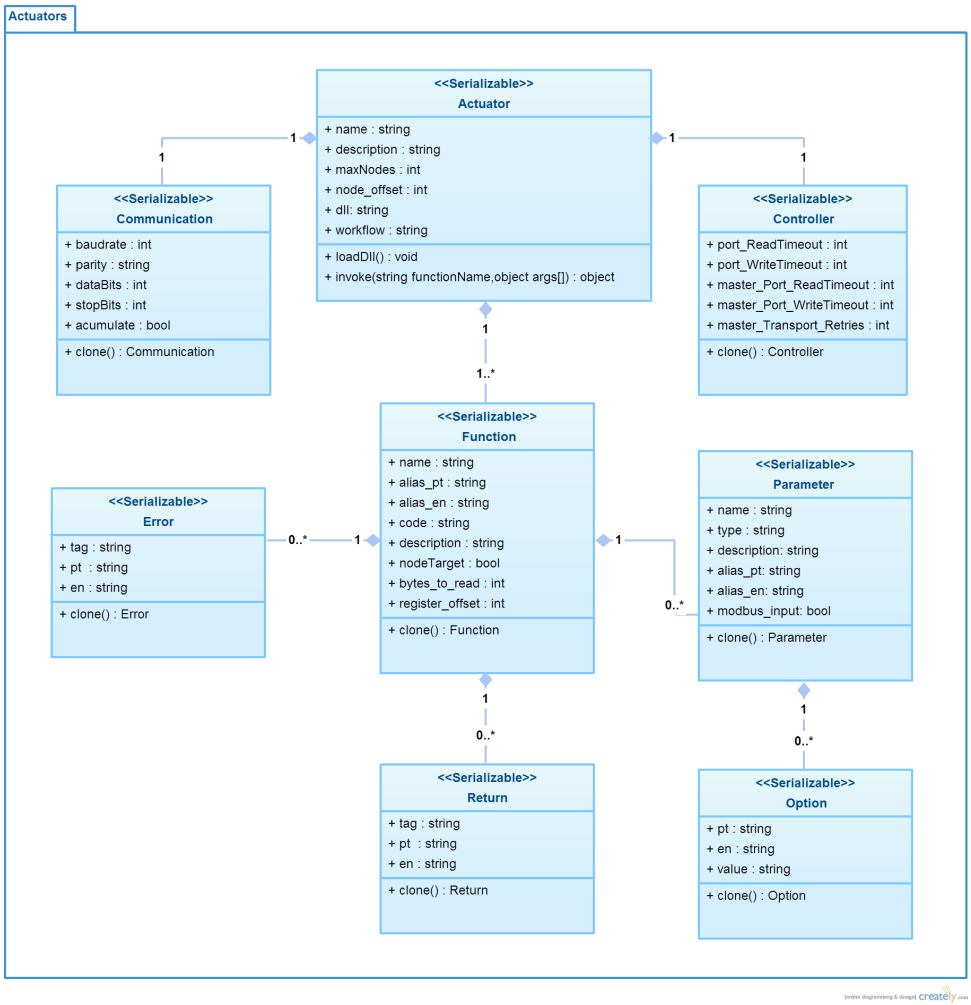

Figura 4.5: Especificação dos atuadores na Framework

no comando Modbus da função respetiva, é possível criar parâmetros auxiliares que poderam ser utilizados no código da função gerada. É tambem possível configurar "opções" para os parâmetros criados. Estas "opções" são úteis quando um determinado parâmetro tem um número limitado de valores. Para cada função é possível determinar se é dirigida a um dispositivo específico (no contexto do esquema Modbus API), ficando esta informação guardada no campo "nodeTarget". Se esta opção for selecionada, o registo destino é calculado em função do "Node Offset" multiplicado pelo ID do dispositivo destino, sendo também possível definir um *offset* adicional que é somado a esta operação ou apenas aplicado caso a função não seja dirigida a um dispositivo específico.

Além dos parâmetros, é possível definir erros e retornos ao criar funções num atuador. Tanto as funções como os parâmetros, erros e retornos possuem diferentes *alias*, que permitem apresentar no *website* a tradução adequada para estes elementos consoante a língua definida. Através da tag do erro definida, é possível enviá-la como resultado de uma certa função, onde será posteriormente apresentado o erro no website através do seu *alias*. A lógica dos erros configurados pelo utilizador deve ser tratada posteriormente no código gerado, uma vez que o método gerado originalmente

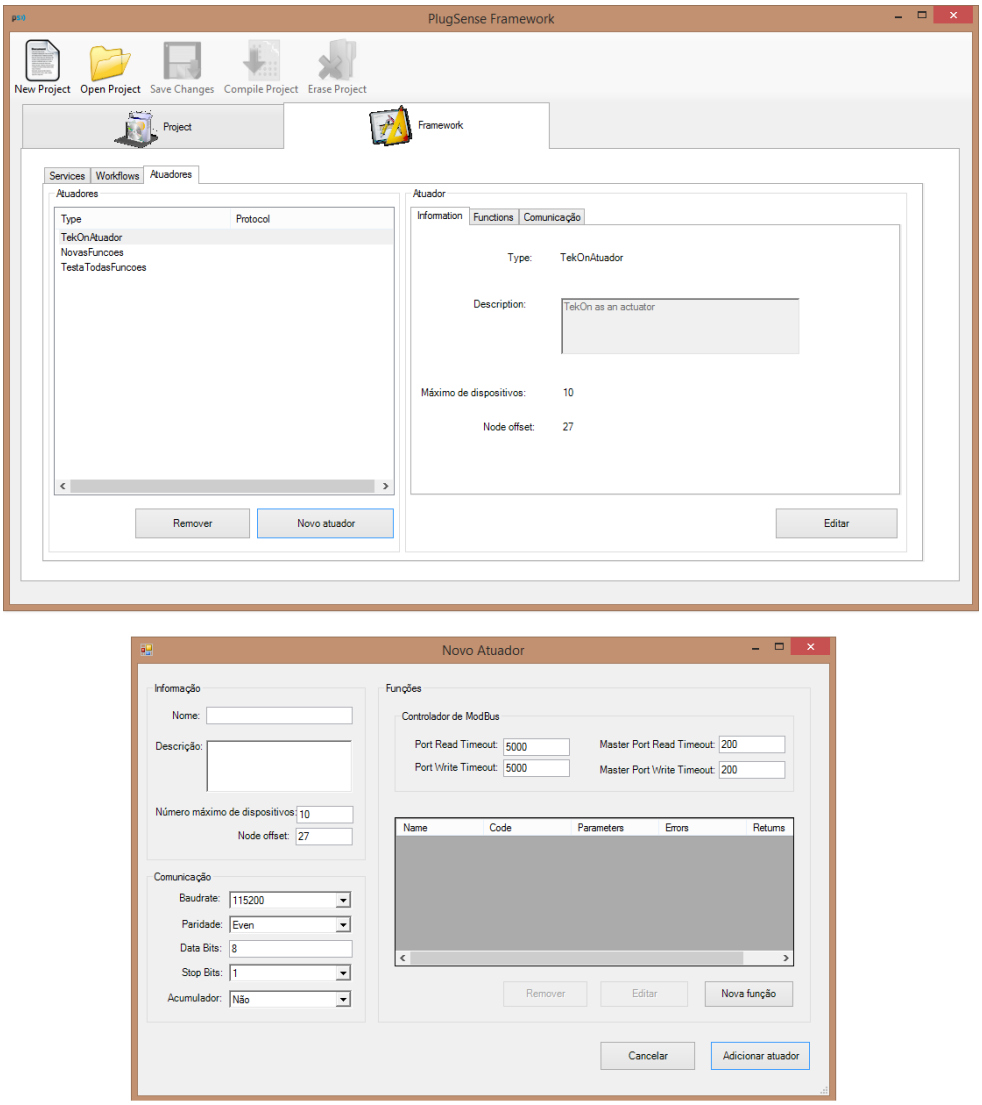

Figura 4.6: Separador de atuadores e janela de novo atuador

apenas trata erros de execução. Os retornos são variáveis que podem ser enviadas como resultado a uma dada função, e que são encapsuladas numa *string* separadas pelo caractere "|" e incluída no objeto de retorno do método. Posteriormente, os retornos são apresentados *website* como o resultado da função executada, identificados pelos respetivos *alias*.

Com estas os componentes parâmetros, erros e retornos é possível adaptar o código gerado para cada função e adicionar toda a lógica que seja necessária para garantir o funcionamento correto do equipamento que se pretende integrar, bem como torná-lo mais robusto.

A grande vantagem na criação de atuadores através da PSF é a geração automática do código da classe que irá ser utilizada na comunicação com o dispositivo na UG. No caso dos sensores, tem de ser previamente e manualmente criado um novo projeto na solução Universal Gateway que contém a classe e os métodos utilizados na comunicação com os dispositivos, e que irá gerar um ficheiro DLL contendo a biblioteca de métodos para os sensores. Na criação de um novo sensor é

então escolhido este ficheiro DLL para que possa ser acedido pela aplicação Universal Gateway e assim invocar o conjunto de métodos necessários para comunicar com os dispositivos. Apesar de haver um *template* com o esqueleto deste tipo de classe, o código dos métodos que inclui tem de ser totalmente reescrito consoante a especificação de cada um dos sensores.

Tendo em conta que o formato das mensagens e funções no protocolo Modbus são pré-definidas e disponíveis em cada conjunto de equipamentos deste tipo, foi possível desenvolver mecanismos que constroem esta classe com todos métodos definidos na interface da aplicação PSF. Desta maneira, em vez de um projeto para cada um dos novos atuadores, foi criado um projeto com o nome ActuatorsGateway que contém as várias classes (com o mesmo nome dado ao atuador) responsáveis pela comunicação dos respetivos tipos de atuadores. Para que a UG tenha acesso a todos métodos destas classes de uma forma modular, foi desenvolvido um *plugin* neste projeto para que no processo de Build seja criada uma biblioteca para cada atuador em vez de criar um ficheiro DLL para o projeto inteiro. Com este mecanismo oferecemos duas grandes vantagens à PSF, uma maior automação na geração de código, bem como uma maior organização no código uma vez que é desnecessário ter projetos individuais para cada tipo de atuador.

Para cada função do atuador definida na PSF é gerado o método com o seu esqueleto no ficheiro da classe com o mesmo nome do atuador, gerando a sua própria biblioteca que irá ser utilizada na execução da UG. Uma vez que se trata de código que irá ser compilado, houve um grande cuidado no controlo dos valores que o utilizar define, como a utilização de caracteres especiais ou espaços, evitando erros de sintaxe que proibiram a classe gerada de compilar. Cada método gerado pela Framework contém as seguintes variáveis de input:

- **int gatewayDevice** — Contém o ID do dispositivo destino.
- **string parameters** — Contém os parâmetros definidos para a função separados pelo caractere "_". Caso a função seja dirigida a um dispositivo específico, o seu ID está presente na primeira posição desta *string*.
- **SerialPort port** — Contém o objeto representativo da porta-série na qual o dispositivo está ligado fisicamente.
- **Semaphore semaforo** — Contém o semáforo que garante que a ocupação da porta-série seja exclusiva durante a transmissão das mensagens.

Em cada classe gerada pela Framework, é também gerado um método "start", que tem como objetivo criar uma instância do controlador Modbus com as configurações definidas no atuador, e que irá ser utilizado para executar a função Modbus na porta-série. Este método é executado em todos os métodos das funções criadas na Framework.

Além do código da classe que é responsável pela comunicação com os dispositivos na UG, é também gerado um *workflow* no projeto Workflows por cada atuador criado na PSF. Cada *workflow* corresponde ao conjunto e sequência de métodos que são executados no servidor ao receber o ficheiro XML enviado pela Universal Gateway após a mensagem ser entregue ao atuador a que

foi dirigido. É também gerado um modelo na PlugSenseApp para cada atuador criado, que tem o objetivo de representar os resultados das ações executadas. Estes modelos têm o nome do atuador e à semelhança dos *workflows*, têm todos a mesma estrutura, podendo esta ser modificada posteriormente para refinar as características e funcionalidades que cada atuador apresenta.

A estrutura dos *workflows* bem como os mecanismos que envolvem o fluxo de comunicação entre a aplicação web e a UG para executar ações nos atuadores e obter a confirmação da sua execução serão abordados após a descrição das entidades e módulos desenvolvidos na PlugSense App para suportar atuadores no sistema, que irão ser apresentados de seguida. Como já foi dito, na Secção 5.3 do próximo capítulo irá ser apresentado um exemplo de integração de um dispositivo Modbus através da Framework desenvolvida, sendo posteriormente apresentados o código da classe e métodos gerados no anexo A.

4.3.2 Utilização de atuadores na aplicação

Terminando a descrição dos módulos responsáveis pela integração de atuadores na PSF, serão abordados os módulos que permitem a interação com os dispositivos através da aplicação web. O processo de registo de atuadores na aplicação é feito de forma manual e por isso distinto do processo de registo dos sensores. No caso dos sensores, ao serem ligados à máquina onde a Universal Gateway é executada, começam automaticamente a enviar dados, que ao serem recebidos do lado do servidor e caso o sensor ainda não exista no sistema, é criada uma nova entrada na tabela de sensores na base de dados que representa este novo sensor no sistema. Como os atuadores não se comportam desta forma e apenas executam funções quando lhes são enviados pedidos, o seu registo tem de ser feito de forma manual. Na Figura 4.7 está representada a especificação dos modelos e tabelas da base de dados que foram criados de forma a representar as entidades formam um atuador.

De seguida é feita uma descrição de cada um destes modelos. As entradas das tabelas da base dados dos primeiros 4 modelos apresentados são inseridos pela aplicação Generator através dos ficheiros XML que contêm toda a informação sobre os atuadores, sendo o seu conteúdo fixo na utilização da aplicação até novos atuadores serem integrados ou haja alterações nos atuadores já existentes. Os restantes são objetos podem ser dinamicamente criados pelo o utilizador da aplicação.

- **ActuatorType** — Representa os tipos de atuadores no sistema. Quando um utilizador deseja registar um novo dispositivo terá de escolher que tipo de atuador quer registar, sendo todos os tipos de atuadores apresentados com base nos dados presentes nesta tabela.

- **ActuatorFunction** — Representa as várias funções que atuadores suportam. Os dados desta tabela permitem exibir que funções o utilizar pode executar consoante o tipo de atuador a que for dirigido.

- **FunctionParameter** — Na tabela deste modelo estão presentes os parâmetros de cada função existente no sistema, que a são presentados no formulário de criação de uma nova ordem

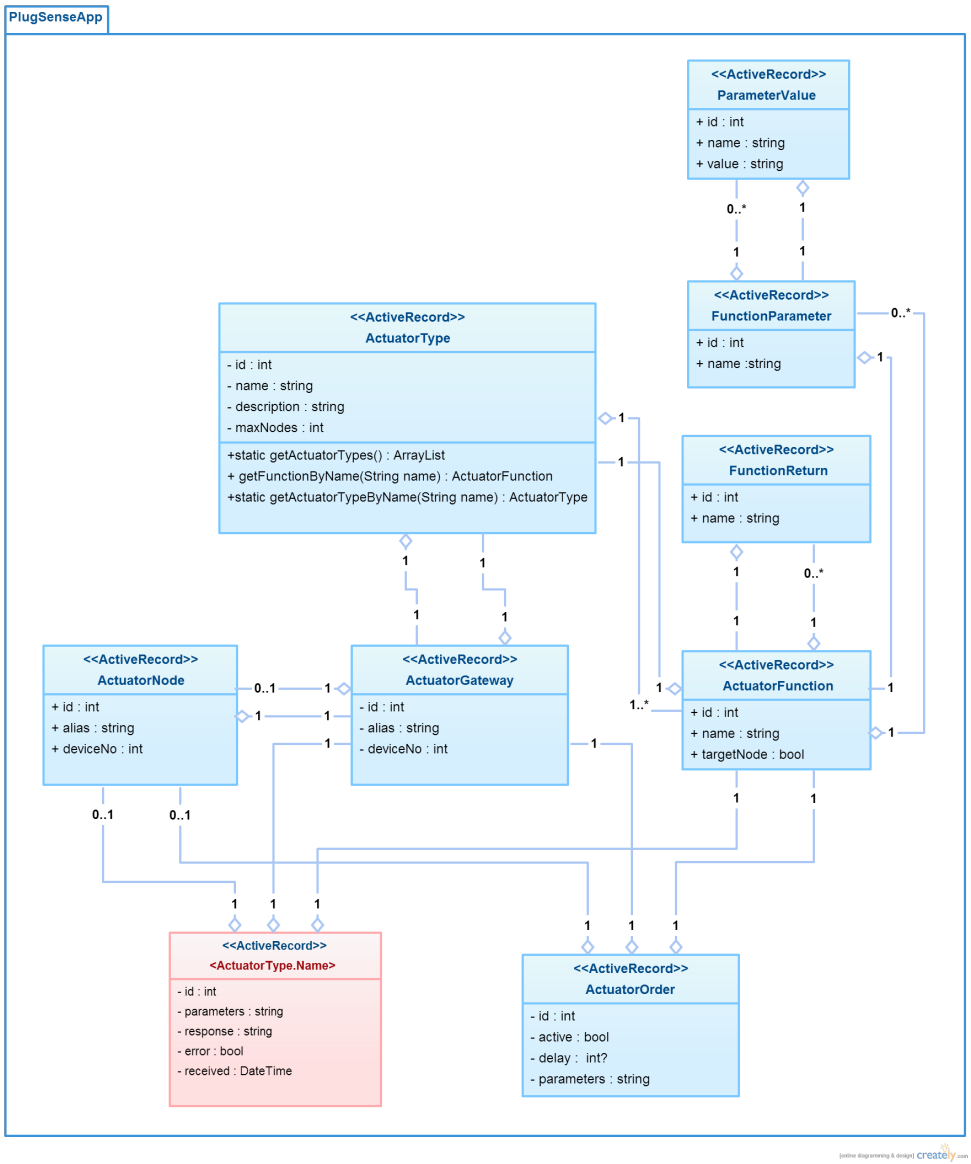

Figura 4.7: Especificação dos atuadores na aplicação web

e de preenchimento obrigatório no momento em que o utilizador deseja criar uma nova ordem a ser enviada ao atuador.

- **ParameterOption** — No caso de algum parâmetro de função ter valores fixos de input, ou seja opções, estes são dados a escolher ao utilizar através do tipo de *input* "select" ou "combo box" no formulário que permite criar uma nova ordem.

- **ActuatorGateway** — Representa um dispositivo ao qual as ordens são dirigidas, e está obrigatoriamente associado a um ActuatorType. Possui um "DeviceNo" que deve ser único entre as *gateways* de um tipo de atuador, pois é este ID que é utilizado para identificar qual o *slave* a que a mensagem se dirige no protocolo Modbus.

- **ActuatorNode** — Representa um nodo do atuador caso o atuador implemente um protocolo Modbus API, e está obrigatoriamente associado a uma *gateway*. À semelhança do "ActuatorGateway", este objeto também tem um "DeviceNo" que deve ser único entre os nodos da *gateway* a que pertence.

- **ActuatorOrder** — Entidade utilizada para representar as ordens que são dirigidas aos atuadores e enviados à Universal Gateway. Estão obrigatoriamente associadas a uma função e uma *gateway* destino. Também pode estar associado a um nodo e conter parâmetros de input caso a função esteja definida com estas componentes.

Além destes modelos, para cada um dos atuadores criados através da Framework é gerada uma classe com o mesmo nome dado ao atuador, que está representado a vermelho na Figura 4.7. Este modelo tem como objetivo representar os resultados das ordens enviadas a um determinado nodo ou *gateway*, e têm portanto uma estrutura fixa quando são gerados. No entanto, é possível a alteração deste modelo a nível de variáveis e métodos para que a representação do funcionamento do dispositivo seja o mais adequado possível. Este conjunto de modelos permite que o utilizador interaja com os atuadores que estão em funcionamento na UG.

Depois de criadas as entidades que representam os dispositivos, o utilizador pode então criar ordens que pode enviar à UG, com variados parâmetros ou dispositivos destino. A entidade ordem é uma grande mais-valia para aplicação pois permite reutilizar certas ações, sem ter de as criar sempre que as deseja executar. Outra particularidade das ordens é que podem ser definidas como periódicas, estando esta opção disponível na página da sua criação. Desta forma, se o utilizador pretender que uma dada ação seja executada periodicamente poderá definir qual será esse intervalo de tempo, e uma vez criada, a ordem pode ser enviadas à UG para que o processo comece, bem como terminá-lo a qualquer momento no caso de uma ordem periódica estar ativa. As ordens podem ser editadas ou removidas consoante o utilizar pretender. O utilizador tem também a opção de enviar pedidos de execução de várias ordens em simultâneo.

Ao enviar ordens na aplicação, a informação é transmitida ao servidor, que as irá compactar num só objeto (string) e utilizar a classe do projeto MessageQueue para as partilhar com as várias instâncias da UG que possam existir. Este objeto contém o tipo de atuador, a função a executar, o ID do dispositivo de *gateway* e nodo caso se aplique na função, parâmetros de input, e os IDs das entradas de base dados que representam a *gateway* e nodo como entidades da aplicação, para que estes possam ser emparelhados com uma determinada porta-série na UG e identificados ao enviar a mensagem de resposta ao servidor.

O passo seguinte é precisamente percorrer todas as instâncias de *webservices* ativas, que correspondem aos vários processos da UG em execução, e em cada uma despoleta um evento sinalizando que estão disponíveis novas ordens para ser executadas ou paradas no caso do cancelamento de uma ação periódica. Existe também um mecanismo no *webservice* que permite enviar às instâncias da UG todas as ordens que são periódicas e estão ativas, para que estas sejam executadas ao iniciar a UG sem intervenção do utilizador.

Uma característica do funcionamento da UG que é necessário referir é que esta possui uma variável responsável pelo mapeamento de todas as portas COM existentes na máquina, associando-as a um tipo de sensor e registando a informação num ficheiro XML. A este objeto de mapeamento iremos chamar "variável de configuração de portas". Na aplicação da UG as portas têm 3 estados possíveis:

- **Aberta** — Uma porta a qual é possível aceder e ainda não está associada a nenhum dispositivo.
- **Fechada** — Uma porta que não é possível aceder, estando provavelmente a ser utilizada por outro processo ou aplicação.
- **Em utilização** — Uma porta que é possível aceder e já está associada a um determinado dispositivo.

Periodicamente é executada uma rotina que testa as portas da máquina disponíveis, no sentido de obter dados que os sensores enviam automaticamente, processá-los e enviados ao servidor. Para cada porta "Aberta" é então percorrida a lista de todos os sensores e com os dados da leitura é executada uma função presente em cada ficheiro DLL dos sensores, que permite identificar se a mensagem recebida pertence ao tipo de sensor em questão. Caso a porta não permita a leitura de dados a porta é considerada como "Fechada". Caso a leitura seja conseguida, é identificado o tipo de sensor, a porta é marcada como "Em utilização" e é enviada uma mensagem do formato PlugSenseML especial que avisa o servidor que foi adicionado um novo sensor. O servidor insere um novo sensor na base de dados e responde à UG o ID atribuído. Ao receber o ID, a UG regista no ficheiro XML de configurações de portas e adiciona à variável de configurações de portas o tipo e ID do sensor. Em cada porta "Em utilização" é executado o método que trata de descodificar os dados e enviá-los ao servidor, e que está disponível na biblioteca do tipo de sensor a que a porta está associada. Se a porta estiver "Aberta" mas nenhum tipo de sensor for reconhecido, a porta é mantida como "Aberta" e poderá ser ocupada com novos dispositivos nas próximas execuções da rotina. É este mecanismo que permite dar ao sistema a facilidade de utilizar os sensores, sendo apenas necessário ligá-los a uma máquina com a UG em execução e automaticamente o sensor é registado e os dados recolhidos podem ser de seguida observados no *website* após associar o novo sensor a uma entidade.

Uma vez que os atuadores não enviam dados através da porta-séria de forma automática como os sensores, foi necessário adicionar mecanismos e estruturas que permitem identificar que um atuador foi ligado a uma porta, registá-la e permitir o futuro fluxo de informação, que no caso dos atuadores pode ser bidirecional.

Ao receber novas ordens, a UG começa por descodificar a mensagens e identificar a que tipo de atuador se dirigem e o número do dispositivo destino. De seguida, e para cada uma das ordens, são percorridas todas as portas-série que estejam já associadas à *gateway* a que ordem é dirigida, e a função é executada através da biblioteca do atuador presente no ficheiro DLL correspondente.

Caso ainda não exista nenhuma porta associada ao dispositivo destino são iteradas todas as portas "Abertas" e a função é invocada da mesma maneira.

Existem vários cenários possíveis ao enviar uma mensagem pela porta-série com o controlador Modbus e consequentemente diversos resultados:

Caso a função não esteja presente na biblioteca de funções do atuador, esta vai levantar um exceção em particular que ao ser apanhada na UG, continua a rotina de teste das portas, pois significa que a porta não está ligada a um atuador do tipo indicado na ordem, mantendo assim a porta "Aberta".

Caso a função esteja presente na biblioteca, o seu código é executado. Uma vez que a porção de código de cada método em que a mensagem é submetida ao controlado Modbus é protegida por um bloco *try-catch*, caso a exceção seja do tipo "Gateway Path Unavailable", significa que o dispositivo associado à porta não corresponde ao dispositivo a que ordem é destinada, forçando assim o levantamento desta exceção para que a rotina da UG ignore esta porta, deixando-a "Aberta" para que possa testar as restantes.

Caso não ocorra nenhuma exceção na execução do código o resultado é sempre retornado à UG num objeto, seja esta com a mensagem de sucesso com possíveis parâmetros de retorno, ou com um dos erros personalizado pelo utilizador. Ao receber este objeto a UG sabe que esta porta está ligada a um atuador deste tipo, e em caso de sucesso associa assim a porta ao tipo de atuador e ID presentes na ordem, registando esta informação no ficheiro XML de configuração de portas.

Se a ordem for periódica é adicionado um *timer* com o período indicado na ordem, para que esta seja executada ao fim desse tempo. Se a ordem tiver indicação para parar esta ordem, o respetivo *timer* é removido.

Continuando o seguimento do fluxo de comunicação entre a aplicação web e UG, depois do método ter sido executado, o próximo passo que a UG toma é codificar a informação de retorno da função no formato PlugSenseML para ser enviada ao servidor, estando este processo contido no projeto SendData da solução Universal Gateway. O formato PlugSenseML teve de ser adaptado ao suporte dos atuadores, uma vez que estes têm uma estrutura diferentes dos sensores. Uma vez contruído o ficheiro XML, este é enviado ao servidor que irá processar o *workflow* do tipo de atuador a que a ordem corresponde.

4.4 Conclusões

Após a descrição detalhada das etapas de desenvolvimento apresentadas neste capítulo é possível perceber as diferentes contribuições do trabalho efetuado nesta dissertação.

As alterações feitas na estrutura da aplicação genérica original da PSF são essências para que as aplicações geradas através da Framework consigam representar um cenário agrícola, onde o esquema de permissões implementado permite uma melhor gestão das parcelas de terreno entre os utilizadores da aplicação. As funcionalidades introduzidas na aplicação genérica são uma mais-valia para a Framework, uma vez que podem ser utilizadas em diversas áreas da agricultura.

A integração da estação meteorológica Vantage Pro2 foi um processo desafiante, pois teve de ser acrescentado um novo módulo à UG, uma vez que o fabricante não permite a comunicação direta com os equipamentos. No entanto, foi possível criar os mecanismos que permitem a aquisição dos dados das estações em tempo real, sendo este equipamento muito útil para agricultura uma vez que inclui um conjunto de sensores capazes de recolher 44 fatores ambientais.

O processo de acrescentar à PSF a capacidade de suportar atuadores foi a etapa mais demorada no desenvolvido desta dissertação, uma vez que para alcançar este objetivo tiveram de ser criados de raiz novas estruturas e mecanismos em vários módulos da PSF. Tendo em conta o resultado final, é possível afirmar que o processo de integração de atuadores Modbus pode ser feito de uma forma ágil, uma vez que a interface gráfica criada permite gerar uma grande parte (e em alguns casos a totalidade) do código necessário para que a comunicação com os dispositivos seja feita corretamente, de acordo com a escificação do equipamento a integrar. As estruturas e mecanismos criados na aplicação web que permitem a interação remota com os atuadores são adequadas para qualquer equipamento Modbus, onde as ordens e a capacidade de as definir como periódicas oferecem as condições de usabilidade desejadas para este tipo de interação.

Tabela 4.1: Funcionalidades e permissões

Operação	Permissões			
	Nenhuma	Parcial	Total	Admin
Ver outros utilizadores do grupo		✓	✓	✓
Criar utilizadores num grupo		✓	✓	✓
Editar dados de outros utilizadores			✓	✓
Excluir utilizadores de um grupo			✓	✓
Alterar permissões num grupo			✓	✓
Adicionar utilizadores a um grupo			✓	✓
Eliminar utilizadores				✓
Criar utilizadores sem grupo				✓
Criar parcelas num grupo		✓	✓	✓
Criar observações		✓	✓	✓
Criar armadilhas		✓	✓	✓
Remover parcelas num grupo			✓	✓
Alterar área da parcela			✓	✓
Editar armadilhas			✓	✓
Editar observações			✓	✓
Remover armadilhas			✓	✓
Remover observações			✓	✓
Editar informação da parcela			✓	✓
Criar grupo				✓
Criar sub-grupo			✓	✓
Remover sub-grupo			✓	✓
Eliminar grupo				✓
Alterar área de um grupo			✓	✓
Associar sensores a um grupo ou entidade			✓	✓
Criar, remover e editar gateways e nodos			✓	✓
Criar, editar e remover ordens			✓	✓
Enviar ordens a atuadores		✓	✓	✓
Parar ordens periódicas		✓	✓	✓
Associar modelo de doença a grupo ou entidade				✓
Arquivar alertas		✓	✓	✓

Capítulo 5

Validação

Neste capítulo será abordado o trabalho efetuado com o objetivo de validar as funcionalidades da Framework desenvolvida e das aplicações que permite gerar.

Na primeira secção serão apresentados os testes elaborados para testar a fiabilidade das funcionalidades acrescentadas à Framework e às aplicações.

De seguida, será apresentado o caso de uso de uma aplicação gerada através da Framework que posteriormente foi adaptada tendo em conta as necessidades da APAS.

Por fim, será apresentado o processo de integração de um dispositivo que implementa o protocolo Modbus API.

5.1 Testes

À exceção das classes que representam os modelos de doenças implementados, não foram desenvolvidos testes unitários nas classes e modelos criados na Framework e na aplicação, uma vez que os métodos que podem alterar o seu estado são efetuados nos controladores da aplicação ASP.NET ou no *webservice* e na interface da Framework.

Assim, houve um especial cuidado na construção da interface, tanto da aplicação como da Framework, no sentido de controlar e não permitir a introdução de dados inválidos por parte do utilizador. O caso de uso que será apresentado na secção seguinte ajudou a refinar algumas falhas que a interface da aplicação apresentava, tendo-se no final adquirido a garantia que todas as funcionalidades lidam com todos os tipos de erros e estão a funcionar corretamente.

No entanto, foi construído um projeto de testes para validar o resultado dos modelos de doenças introduzidos na aplicação, garantido que os cálculos sejam corretamente executados. Este projeto foi criado na solução Universal Gateway, utilizando métodos públicos do projeto principal para a aquisição de dados que servem de input para os métodos de teste, contendo uma instância do *webservice* onde os modelos são executados.

Os dados pertencem a um conjunto de ficheiros exportados pelo WL, cedidos pela APAS e pertencentes a uma das estações que possuem, correspondendo aos dados recolhidos durante o ano 2013. Após o processamento dos ficheiros, os ficheiros XML resultantes são enviados através

de um método especialmente criado no *webservice* para a execução de testes. Este método cria uma parcela fictícia a qual associa todos os modelos de doença disponíveis com os vários sensores da estação a qual os dados pertencem. De seguida, os processos do *workflow* da estação são executados normalmente, sendo os modelos calculados após os dados serem guardados na base de dados.

Por fim, o resultado dos modelos é comparado com os resultados presentes nas folhas de cálculo correspondestes à estação de testes e cedidas pela APAS. No caso dos resultados numéricos, uma vez que alguns dos valores resultantes dos cálculos apresentam um elevado número de casas decimais, foi utilizada uma margem de erro de 0.005 para que os resultados fossem aceites.

Uma vez terminados os testes, o seu resultado é enviado à classe de testes através do mecanismo MessageQueue, eliminando previamente da base de dados todos os objetos associados à parcela fictícia criada, de forma a manter a base de dados da aplicação no seu estado inicial.

Desta forma foi possível verificar que os cálculos executados nas classes dos modelos de doenças oferecem resultados de acordo com a especificação apresentada na Secção 2.6, durante as várias alterações que sofreram nas iterações do desenvolvimento do projeto.

5.2 Caso de uso

A colaboração da APAS no âmbito desta Dissertação teve início com a necessidade que a associação tinha em inovar o sistema que tinha em funcionamento, que consiste em fornecer folhas de cálculo com a informação dos modelos de prevenção de doenças apresentados na Secção 2.6 aos agricultores em forma de serviço. De seguida será apresentado o processo que a associação mantinha em funcionamento, e o qual foi renovado pela utilização de uma aplicação gerada através da Framework desenvolvida, e posteriormente adaptada.

Como já foi referido na Secção 4.2, a APAS dispõe de um conjunto de estações meteorológicas Vantage Pro2 instaladas por diferentes regiões do centro de Portugal. Através do *software* WL, os dados de cada estação eram manualmente transferidos e exportados para um ficheiro no formato TXT. Com o mecanismo *copy & paste*, os dados eram introduzidos numa folha de cálculo XLSX na qual estão incorporados os cálculos dos modelos referidos. Este ficheiro servia então como *template* para efetuar os cálculos, existindo um exemplar por cada estação, sendo posteriormente partilhados com os agricultores através da plataforma Dropbox. Este processo era repetido aproximadamente duas vezes por semana, mantendo os agricultores o mais atualizados possível.

A partir da Framework desenvolvida foi então criada uma aplicação com o objetivo de melhor este sistema nos seguintes aspetos:

- Automatizar do processo de transferência e exportação dos dados das estações e cálculo dos modelos de prevenção de doenças em tempo real.
- Automatizar a partilha dos resultados dos modelos de prevenção de doenças.
- Apresentar os resultados dos modelos de uma forma mais intuitiva.

A aplicação proporciona não só a automatização da recolha de dados e execução dos cálculos dos modelos científicos através dos mecanismos já abordados nas secções 4.1 e 4.2, como oferece um *website* de fácil acesso aos utilizadores, onde podem representar e gerir as suas parcelas de terreno, consultar o risco a que as colheitas estão sujeitas em relação à contração das doenças referidas, entre outras funcionalidades.

Na Figura 5.1 encontra-se um diagrama que representa as entidades, modos de comunicação e topologia do sistema das estações, bem como onde se inserem os componentes de integração destes dispositivos na aplicação. Uma vez que o computador central que irá fazer pedidos de dados às várias estações está muito longe dos DLs e estes estão colocados perto das estações que normalmente se situam em terrenos sem ligação à Internet ou rede telefónica, o modo de comunicação utilizado pela APAS e testado nesta solução foram os Modems GSM.

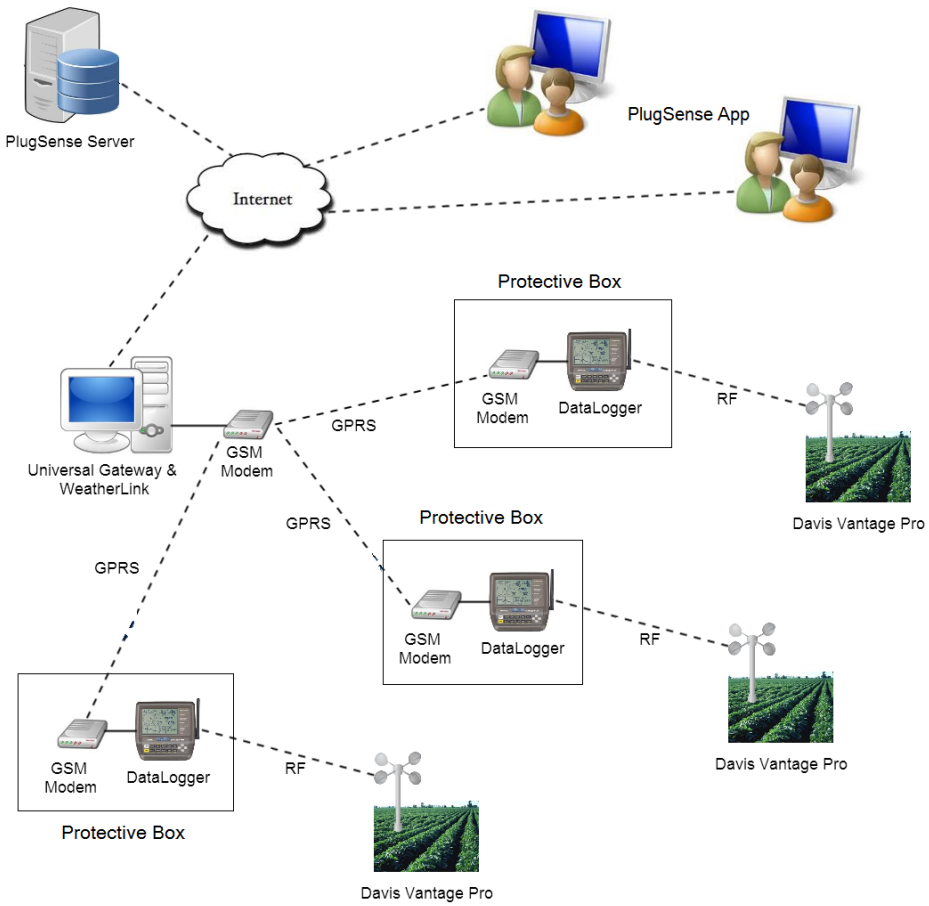

Figura 5.1: Representação do sistema de estações Vantage Pro2 no sistema da APAS

A primeira etapa na modificação da aplicação genérica da Framework foi adaptar a UG para que esta fosse mais fácil de utilizar nas instalações da APAS em relação à versão atual da UG. Uma vez que só irão trabalhar com este tipo de sensores, a interface que permite interagir com os restantes tipo de sensores e atuadores foi ocultada, e a que permite interagir com as estações foi

melhorada, como pode ser observado na Figura 5.2. Através da configuração de cada estação no WL, o processo de transferir e exportar os dados passa a ser automático, podendo esta operação ser feita a cada 15 minutos, que é o tempo entre leituras dos sensores das estações. Definindo qual o ficheiro destino de cada estação na UG, a cada atualização dos ficheiros dos dados são enviados para o servidor, podendo os novos dados da estação e os resultados das doenças ser observados logo de seguida no *website* pelos utilizadores.

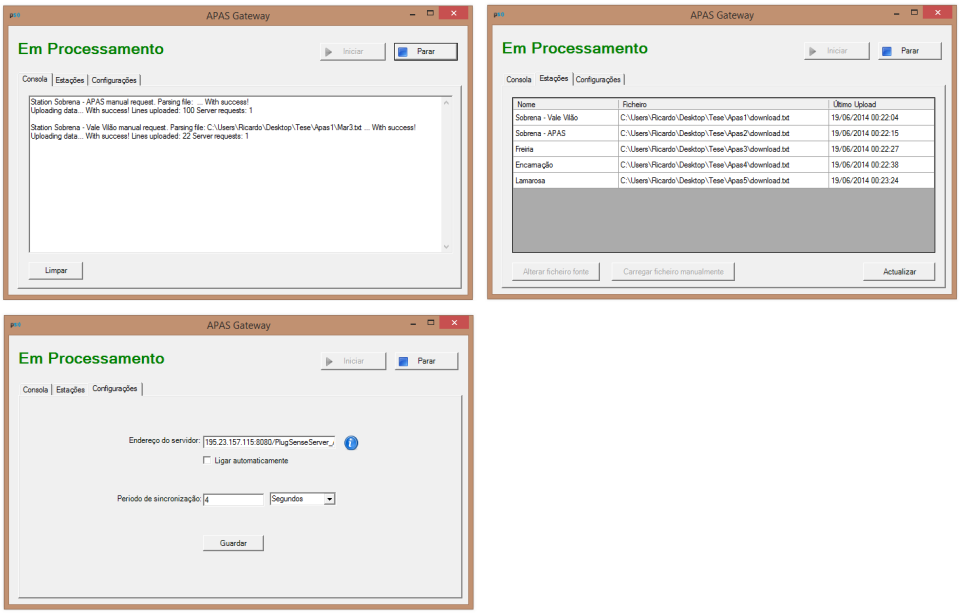

Figura 5.2: Versão da Universal Gateway para a aplicação da APAS

Na aplicação web foram também feitas alterações de forma a melhorar a experiência de utilização, desde aos técnicos da APAS aos agricultores. Uma vez que as estações têm um papel vital neste sistema, foi criado no painel lateral uma lista com as estações acessíveis ao utilizador, que direcionam para uma página dedicada a estações. Nesta página (apresentada na Figura 5.3) podem ser observados os dados recolhidos e as configurações, ao contrário do funcionamento genérico de sensores onde os dados são observados na página da entidade a que estão associados. Foi também criado na secção de administração um separador dedicado à gestão de estações.

Foram feitas alterações nas tabelas de apresentação dos modelos de doenças. Na versão genérica desenvolvida, cada modelo de doença associado a uma entidade é apresentado na página da entidade num separador dedicado, onde pode ser observado através de um gráfico ou tabela. Nesta aplicação, os dados calculado através das leituras das estações são apresentados num único separador com o título "Dados da estação meteorológica", na página do grupo ou parcela. As tabelas dos vários modelos foram agregados numa só tabela, sendo mantidos no entanto separadores para visualização dos modelos em gráfico. Os gráficos com o termopluviograma e o gráfico de humidade do solo também estão contidos neste separador. Além das tabelas dos modelos de doenças, foi também acrescentado um separador com o título "Dados meteorológicos" que contém uma

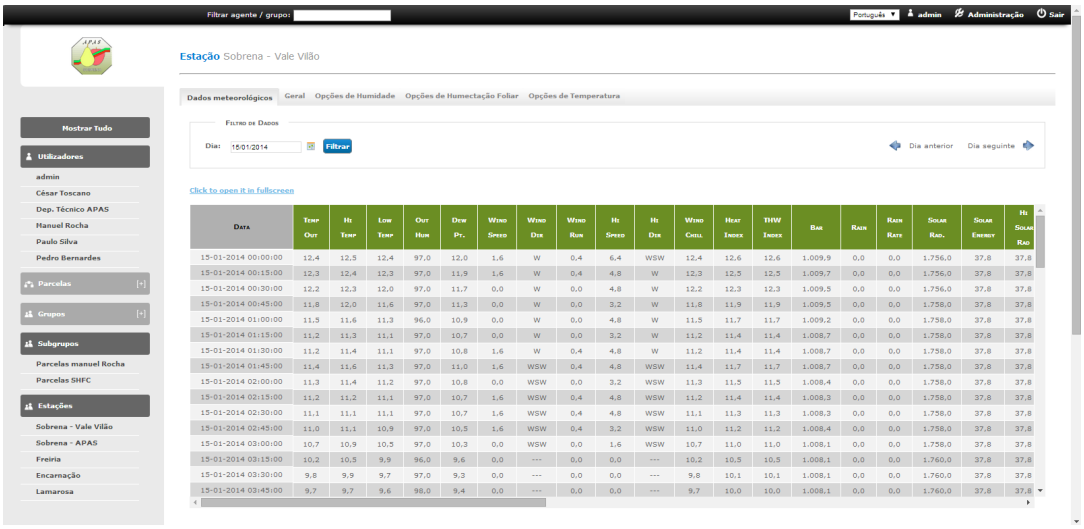

Figura 5.3: Página da uma estação meteorológica

tabela com certas leituras dos sensores, mas com as médias diárias e outro "Zona de cálculo" com alguns valores utilizados cálculos dos modelos de prevenção de doenças. Os valores apresentados nestas tabelas foram selecionados pela APAS.

As restantes funcionalidades da aplicação são idênticas à aplicação genérica que a Framework desenvolvida permite criar, e que foram abordadas no Capítulo 4. Assim a APAS conseguiu não só melhorar o sistema em vigor, como proporcionar uma gestão de parcelas aos agricultores e um conjunto de funcionalidades adicionais, como a visualização do termopluviograma, gestão de armadilhas de insetos, etc.

5.3 Simulação da integração um atuador

Como já foi referido na Secção 4.3, não foi possível adquirir um atuador de forma a integrá-lo na Framework através dos mecanismos desenvolvidos. No entanto, a FreedomGrow dispõe de um exemplar de um sistema de sensores de temperatura que funciona sobre o protocolo Modbus. Desta forma, foi possível testar os mecanismos de geração de código das classes que tratam da comunicação entre a UG e os dispositivos, sendo o comportamento deste muito semelhante à de um atuador.

O equipamento utilizado é fabricado pela empresa portuguesa TekOn Electronics, especializada em transmissores de temperatura digitais, sensores e sistemas wireless. O sistema consiste num dispositivo *gateway* TekOn WGW1104[1] que permite comunicar com um máximo de 16 transmissores de temperatura THUW1103, também fabricados pela TekOn. Os transmissores possuem na sua caixa protetora um sensor de temperatura, comunicando à *gateway* as medições efetuadas através de RF.

[1] http://http://tekonelectronics.com/pt/wgw1104-wireless-modbus-gateway

Figura 5.4: Tabelas e gráficos dos modelos de prevenção de doenças

A *gateway* tem como objetivo configurar o intervalo de tempo com que os transmissores devem efetuar as medições de temperatura, bem como enviar-lhes pedidos para que estes enviem a última medição registada. Desta forma, este sistema de comunicação obedece ao padrão *master/slave*, onde a *gateway* se apresenta como dispositivo *master*, e os transmissores comportam-se como *slaves*. No entanto, a *gateway* apenas envia comandos aos transmissores após lhe terem sido pedidos por um dispositivo externo, normalmente um computador. A comunicação entre a *gateway* e o computador é efetuada sobre o protocolo Modbus através da interface RS485, onde a *gateway* desempenha o papel de *slave* e o computador de *master*. Assim, este sistema pertence ao grupo Modbus API apresentado na Secção 2.3.2, onde a *gateway* oferece uma API para que através de comandos básicos do protocolo Modbus, o computador possa controlar os vários nodos (transmissores) do sistema, comunicando apenas com a *gateway*.

A *gateway* permite a execução de duas funções do protocolo Modbus, onde através da sua tabela de registos e das regras definidas pela API demonstrada na Tabela 5.1, é possível ler um

conjunto de parâmetros como as leituras dos sensores mais recentes ou a tensão da bateria dos transmissores, bem como definir alguns destes parâmetros como o intervalo de tempo de cada medição de temperatura. Para as leituras é utilizada a função "0x04 - Read holding register", e para as escritas é utilizada a função "0x10 – Write Multiple registers". Os vários parâmetros dos nodos são posicionados em posições de memória contíguas da tabela de registos da *gateway*, tendo cada parâmetro um tamanho específico. Uma vez que todos os nodos possuem o mesmo número de parâmetros, é possível aceder ao endereço incial do conjunto de parâmetros de um determinado nodo multiplicando um offset de 27 bytes ao seu ID.

Tabela 5.1: Modbus API para o sensor TekOn

	Temperatura do sensor (°C * 100)	Intervalo de transmissão (s)	Tipo de sonda do sensor	Modelo do nodo	Tensão da bateria (Volt * 100)	RSSI (dBm)	Tempo de vida da informação (s)	Temperatura da caixa (°C * 100)
Tipo de variável	Int32	UInt32	UInt16	UInt16	UInt16	UInt16	UInt32	Int16
Endereço	Offset	Offset + 18	Offset + 20	Offset + 21	Offset + 22	Offset + 23	Offset + 24	Offset + 26
Permissões	Leitura	Leitura e Escrita	Leitura e Escrita	Leitura	Leitura	Leitura	Leitura	Leitura
Valores permitidos	Dependente do tipo de sensor	0 a 86400	1 a 99	1 a 99	3.00 a 20.00	70 a 180	0 a 86400	-20.00 a 80.00

Tendo em conta esta especificação, foi criada através da interface e mecanismos da Framework para a integração de atuadores, um novo tipo de atuador que conseguisse suportar todas as funcionalidades deste equipamento. Para isso, apenas foi necessário criar duas funções, nomeadamente uma que permitisse definir o intervalo de tempo entre medições de temperatura de cada nodo, e outra que efetuasse a leitura da última medição de temperatura e dos restantes atributos. Uma vez gerado o código, foi possível registar na aplicação dispositivos *gateway* deste novo tipo de "atuador", bem como os nodos presentes no sistema, sendo o utilizador capaz de criar ordens que executem estas funções possam interagir com os dispositivos do sistema remotamente. Na Figura 5.5 é apresentada uma visualização gráfica de duas partes da interface que permitiram configurar estas duas funções, tendo sido já apresentada na Figura 4.6 algumas configurações gerais deste "atuador" que foi integrado.

O código gerado para função "DefineInterval" não necessitou de qualquer alteração para que executasse as tarefas pretendidas. Pelo contrário, embora código gerado para a função "GetTemperature" contenha já a estrutura necessária para que a função Modbus seja executada corretamente, existe alguma lógica que obrigou o desenvolvimento de código adicional, nomeadamente o processamento da informação recebida e potenciais erros que possam ter ocorrido. No anexo A é apresentado o código gerado pela Framework para a classe deste novo atuador no projeto ActuatorsGateway. Através da comparação entre o excerto original e a versão final do método "GetTemperature", bem como da observação do método "DefineInterval" totalmente gerado pela Framework, é possível observar o valor que a Framework oferece no processo de integração de atuadores baseados em Modbus, uma vez que a interface se apresenta como uma ferramenta que

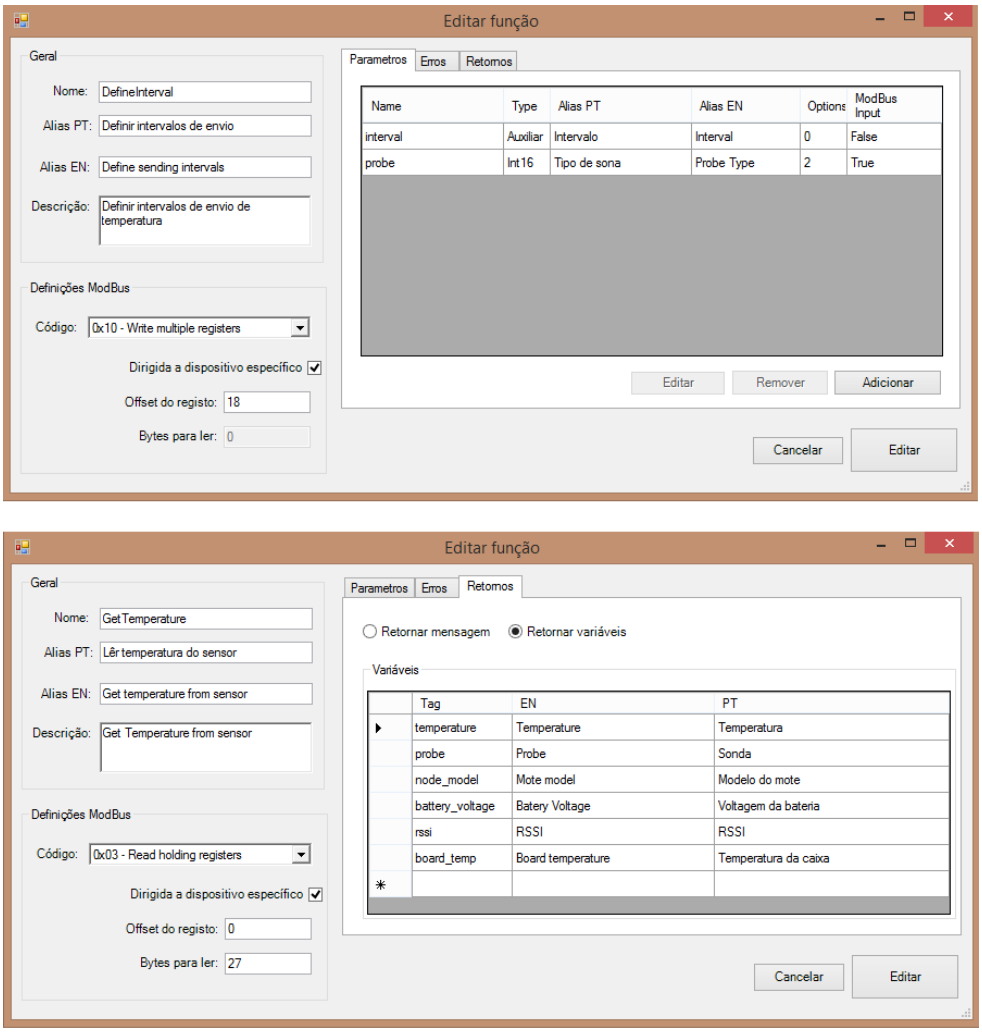

Figura 5.5: Interface da Framework para geração de funções

permite evita o desenvolvimento da maior parte do código necessário à comunicação com os dispositivos, bem como as funcionalidades existentes na aplicação que permitem a interação entre dispositivos e utilizadores.

Uma vez gerado e adaptado o código da classe correspondente a este "atuador", foram utilizados uma *gateway* e um transmissor (nodo) TekOn, com os IDs 0 e 5 respetivamente, de forma a criar um ambiente de teste e validar o funcionamento das funções criadas, bem como a interação entre a aplicação e a UG. Após terem sido criadas as entidades na aplicação, foram também criadas duas ordens distintas para as duas funções do "atuador". Uma das ordens foi configurada com a função "GetTemperature", e periódica com um intervalo de 15 segundos. Desta forma, a UG executa a função no nodo em questão periodicamente, enviado o seu resultado ao *webservice* para que este guarde a informação em base de dados e a disponibilize ao utilizador na aplicação. Foi criada outra ordem com a função "DefineInterval", para que os nodos comecem a efetuar as medições de temperatura também com um intervalo de 15 segundos. Na Figura 5.6 são apresentadas

duas páginas do *website* contendo as ordens criadas e a visualização dos resultados obtidos, com a particularidade de que esta última tabela foi alterada para que apenas demonstre a temperatura recolhida.

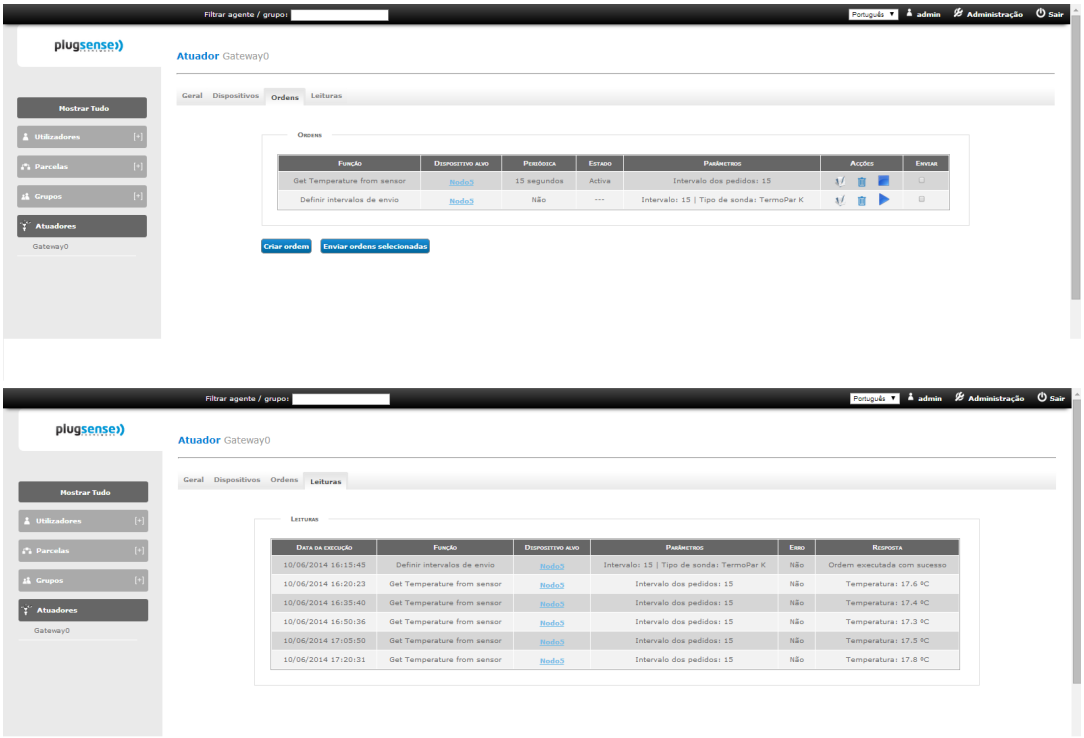

Figura 5.6: Ordens e leituras na integração de um atuador

Capítulo 6

Conclusões

Tendo em conta as falhas de flexibilidade e extensibilidade apresentadas nas aplicações abordadas no Capítulo 3, é possível concluir que a Framework desenvolvida surge como solução a estas dificuldades, tendo sido alcançados com sucesso os objetivos propostos nesta dissertação.

Estas aplicações foram construídas de raiz, o que se traduz num elevado período de desenvolvimento, desde a fase de especificação à fase de conceção. Pelo contrário, as aplicações geradas a partir da Framework apresentam uma estrutura pré-definida e bem especificada, onde milhares de linhas de código são disponibilizadas entre os diferentes módulos das aplicações, nomeadamente a aplicação web, o servidor e a Universal Gateway. Estes módulos oferecem todo o suporte necessário à interação com sensores e atuadores, a nível de estrutura e funcionalidades.

Além de terem sido criadas de raiz, a maioria das aplicações analisadas foram desenvolvidas no sentido de oferecer uma solução a um problema específico, ou seja, a sua utilização noutro contexto da agricultura poderá não preencher os requisitos desta nova área, tanto a nível de funcionalidades como nas tecnologias utilizadas. A Framework propõe uma nova metodologia no desenvolvimento deste tipo de soluções, pois permite gerar aplicações genéricas com as funcionalidade básicas de suporte a sensores e atuadores, que podem ser agilmente adaptadas para cumprir requisitos específicos de um cenário agrícola específico, mantendo no entanto a sua estrutura principal.

O caso de uso da aplicação desenvolvida para a APAS demonstra a utilidade da Framework no desenvolvimento ágil de aplicações para a AP, uma vez que as adaptações feitas na aplicação genérica para solucionar as suas necessidades foram mínimas e concebidas em poucas semanas, produzindo muito pouco código adicional. O feedback recebido pelos utilizadores da aplicação foi muito positivo e houve uma preocupação em oferecer o máximo de usabilidade no website, tendo em conta que muitos dos utilizadores finais não têm conhecimentos informáticos muito elevados. A aplicação estará em fase de produção muito brevemente, embora possam vir a ser acrescentadas novas funcionalidades no futuro.

As aplicações geradas pela Framework são consideradas genéricas no sentido de não conter funcionalidades muito específicas a apenas um ramo da agricultura e ao mesmo tempo oferecer modelos e estruturas que permitam representar entidades comuns num ambiente agrícola, como as parcelas ou as armadilhas de insetos. No entanto, apesar de serem genéricas oferecem um conjunto de funcionalidades muito úteis além da interação com sensores e atuadores, nomeadamente a visualização dos modelos de prevenção de doenças, a georreferenciação dos terrenos, as anotações, a visualização de gráficos auxiliares como termopluviograma, entre outros.

Voltando ao Estado da Arte, nas aplicações revistas a especificação do software que suporta a utilização dos sensores e atuadores não é fornecida, sendo difícil de determinar a possibilidade ou esforço a nível de desenvolvimento que implica a integração de novos equipamentos nos sistemas. Além disso, o conjunto de sensores e atuadores utilizados é restrito, e em alguns casos nem são especificados quais os equipamentos utilizados. Nenhuma das soluções apresenta mecanismos ágeis para integrar novos equipamentos, tanto sensores como atuadores, acusando assim falhas de extensibilidade nos sistemas. Apesar desta funcionalidade se encontrar disponível na Framework antes do início desta dissertação, é possível integrar novos sensores através de uma interface intuitiva, sendo apenas necessário desenvolver o código que permite comunicar com os dispositivos consoante o protocolo e estrutura de dados que apresentam. Após este desenvolvimento adicional, as restantes funcionalidades das aplicações estão preparadas para suportar os novos equipamentos.

A integração das estações Vantage Pro2 provou a utilidade na Framework no que diz respeito à integração de novos sensores nas aplicações. No entanto, foi necessário acrescentar um novo módulo à UG para que esta consiga adquirir os dados das estações através do software oferecido pelo fabricante e dos ficheiros que este exporta contendo os dados dos sensores. Após este módulo estar concluído, todos os mecanismos que são executados a partir do momento em que os dados são enviados pela UG foram gerados a partir da interface da Framework que permite integrar novos sensores, podendo observar os dados recolhidos em tempo real no website automaticamente. O acréscimo deste módulo demonstra mais uma vez a capacidade de estender os vários módulos da aplicação genérica de forma a suportar novos dispositivos e funcionalidades. Além disso, também demonstra a flexibilidade oferecida às aplicações, uma vez que agora é possível utilizar este novo tipo de sensor em todas as aplicações que já tivessem sido geradas pela Framework.

Um dos grandes objetivos propostos nesta dissertação e que foi alcançado com sucesso consistiu em acrescentar à Framework e à aplicação genérica a capacidade de interagir com atuadores, uma vez que este é um requisito essencial nas aplicações dirigidas à AP, e a versão original da Framework e as aplicações que permitia gerar não cumpriam. Para isso foram criados novos módulos, estruturas e mecanismos que permitem representar e interagir com este tipo de equipamentos. A criação de ordens que permitem executar uma determinada função do atuador incluindo um conjunto de parâmetros adicionais e dispositivos destino, oferece uma grande usabilidade na medida que permite a reutilização de comandos, uma vez que não é necessário configurar cada tarefa cada vez que a se pretende executá-la, sendo para esse efeito apenas necessário efetuar um *click*.

À semelhança do mecanismo de integração de sensores, a foi também acrescentada à Framework um conjunto de ferramentas que permite integrar novos atuadores agilmente através de

uma interface intuitiva, de forma a poder utilizá-los em novas aplicações ou em aplicações já existentes. Apesar de apenas ser possível integrar atuadores baseados em Modbus, esta é uma tecnologia de comunicação muito utilizada em atuadores na AP. Através da interface da Framework é possível criar tarefas que executam as funções básicas do Modbus, mas também podem incluir elementos personalizados, como parâmetros adicionais ou erros personalizados, oferecendo assim condições de flexibilidade para integrar dispositivos com funcionamentos específicos.

A Freedom Grow não conseguiu adquirir um atuador de forma a poder integrá-lo na Framework através das componentes desenvolvidas, e ser uma opção válida no desenvolvimento de novas aplicações finais ou em aplicações existentes. No entanto, o grande objetivo foi cumprido, que consiste em permitir a integração de forma ágil de atuadores com auxílio de uma ferramenta que facilita o processo de desenvolvimento. Assim, uma vez integrado um novo atuador através da Framework, é gerado o código base que permite a comunicação direta entre a UG e os dispositivos. Mesmo que seja necessário efetuar uma refinação deste código consoante a especificação de cada atuador, é possível utilizar este tipo de atuador nas aplicações finais, não sendo necessário efetuar alterações nos restantes módulos das aplicações, nomeadamente o *website*. Apesar de não ter sido integrado um atuador real, o ambiente de teste com o sensor TekOn permitiu validar os mecanismos de integração de atuadores, tanto a nível de execução como de utilidade.

Em suma, os objetivos propostos para o trabalho desta dissertação foram concretizados com sucesso. A Framework desenvolvida apresenta-se como uma verdadeira ferramenta para criar aplicações dirigidas à AP, simplificando em vários níveis o processo de desenvolvimento. A aplicação genérica contém uma estrutura e um conjunto de funcionalidades capazes de satisfazer vários cenários agrícolas, e a característica de poder gerir as aplicações finais oferece as condições flexibilidade propostas nesta dissertação, podendo ser alteradas sem prejudicar o seu funcionamento atual. As tecnologias Microsoft que suportam a Framework são acessíveis, permitindo alterar ou acrescentar novos módulos como o foi o exemplo do acréscimo de suporte a atuadores que antes do início desta dissertação não era possível, oferecendo também as condições de extensibilidade desejadas. A Framework poderá não só ser uma ferramenta valiosa para a FreedomGrow no desenvolvimento de aplicações finais à medida e a pedido de clientes, mas também por profissionais que desejem desenvolver as suas próprias aplicações e integrar novos sensores e atuadores com os mecanismos que a Framework oferece.

6.1 Trabalho futuro

Apesar da Framework desenvolvida permitir criar aplicações dirigidas à AP com funcionalidades úteis e com suporte de determinados sensores e atuadores, existe ainda trabalho que pode ser desenvolvido no sentido de evoluir as capacidades da Framework de forma a criar aplicações mais robustas e com mais funcionalidades.

No trabalho desenvolvido apenas foi integrado um novo tipo de sensor na Framework. Embora este novo tipo de sensor tenha aumentado o leque de sensores que a PSF suporta e recolha dados ambientais muito úteis na AP, deverá haver um esforço para integrar mais sensores que se

possam adaptar a outros contextos, uma vez que as Vantage Pro 2 são adequadas para culturas *outdoor*, como pomares por exemplo. Tendo em conta a agricultura efetuada em estufas, a aplicação também deverá apresentar um mecanismo para representar as parcelas existentes, uma vez que a georreferenciação não é útil neste contexto.

Os modelos de doenças disponibilizados na aplicação foram sugeridos pela APAS uma vez que são os mais frequentes na cultura de peras e maçãs e que são possíveis de calcular através dos sensores integrados na PSF. No futuro poderão ser adicionado modelos de previsão de outras doenças comuns noutro tipo de colheitas. Além disso, seria uma mais-valia poder calcular os modelos com base em previsões de dados ambientais, através de serviços como o WindGuru, por exemplo.

Como foi explicado na Secção 4.3 não foi possível integrar na Framework um atuador real, sendo este um passo essencial para que as aplicações criadas através da Framework possam ser atrativas num cenário agrícola que exija a utilização de atuadores. Uma vez que esta versão da Framework irá ser continuada pela Freedom Grow, os contactos com fornecedores de atuadores irão continuar mantendo o foco em sistema de irrigação, podendo no entanto ser integrados outros tipos de atuadores, consoante a procura deste produto se manifestar. Para que a Framework apresente as condições de extensibilidade desejadas, será também necessário incluir na Framework os mecanismos que permitem a integração de atuadores que funcionem sobre outras tecnologias de comunicação, uma vez que o trabalho desenvolvido apenas permite integrar dispositivos baseados no protocolo Modbus.

Uma tarefa que poderá elevar as capacidades da Framework será desenvolver estruturas e mecanismos que permitam definir a execução de tarefas por parte dos atuadores baseados num conjunto de regras definidas sobre condições dos valores recolhidos pelos sensores, formando assim um sistema de apoio à decisão. Seria também uma funcionalidade poderosa e inovadora permitir que os utilizadores definam estas regras, ou seja, que configurem os valores limite para séries dos sensores em particular e determinar que ações um certo atuadore deve executar em caso de alerta.

Embora a PSF possua manuais de utilização (tanto da Framework, como da UG e da aplicação), será necessário adaptá-los, introduzindo instruções adequadas ao funcionamento geral dos diferentes módulos da Framework tendo em conta o desenvolvimento levado a cabo nesta dissertação. Apenas desta forma será possível que a Framework desenvolvida esteja totalmente apta para ser adquirida e utilizada em grande escala, uma vez que este é um aspeto que influencia bastante o grau de satisfação do utilizador.

Referências

[ADB+99] G. Abowd, A.K. Dey, P. Brown, N. Davies, M. Smith e P. Steggles. Towards a better understanding of context and context-awareness. *The Workshop on TheWhat, Who, Where,When, and How of Context-Awareness as part of the 2000 Conference on Human Factors in Computing Systems, The Netherlands*, pages 304–307, 1999.

[AIS11] Abu Zafar Abbasi, Noman Islam e Zubair Ahmed Shaikh. A review of wireless sensors and networks' applications in agriculture. *Computer Standards & Interfaces*, pages 1–8, 2011.

[aSYN+10] Zubair a. Shaikh, Humaira Yousuf, Farah Nawaz, Muneebah Kirmani e Sara Kiran. Crop irrigation control using wireless sensor and actuator network (wsan). *International Conference on Information and Emerging Technologies*, pages 1–5, 2010.

[CdS09] José Pimentel Castro Coelho e José Rafael Marques da Silva. Agricultura de precisão. *Inovação e Tecnologia na Formação Agrícola*, 2009.

[CLea09] Y.H. Cai, G. Liu e L. Li et al. Design and test of nodes for farmland data acquisition based on wireless sensor network. *Chinese Society of Agricultural Engineering 25(4)*, pages 176–178, 2009.

[Cor96] Echelon Corporation. Lonworks twisted pair control module user's guide, version 2. disponível em http://www.echelon.com/support/documentation/manuals/transceivers/078-0015-01F_LWTPCMUG.pdf, 1996.

[CZ06] X. Chen e F. Zhang. The establishment of fertilization technology index system based on "3414" fertilizer experiment. *China Agricultural Technology Extension 22*, pages 36–39, 2006.

[eTHN] Centro Operativo e Tecnológico Hortofruticola Nacional. Proteção de culturas. disponível em http://infoagro.cothn.pt/portal/index.php?id=2270.

[FG06] P. Fuhrer e D. Guinard. Building a smart hospital using rfid technologies. *1st European Conference on eHealth Fribourg, Switzerland, 12-13 October 2006, Fribourg, Switzerland*, pages 1–14, 2006.

[FZG07] Y.B. Feng, R.B. Zhang e G.D. Gu. Application of wireless sensor network in water-saving irrigation. *China Rural Water and Hydropower (2)*, pages 24–26, 2007.

[GL92] Joaquín García e Otazo López. *Peral: control integrado de plagas y enfermedades*. Agro Latino, 1992.

[HWH+10] J. He, J. Wang, D. He, J. Dong e Y. Wang. The design and implementation of a integrated optimal fertilization decision support system. *Mathematical and Computer Modelling 51*, pages 155–168, 2010.

[KSD+10] D. Kolokotsa, G. Saridakis, K. Dalamagkidis, S. Dolianitis e I. Kaliakatsos. Development of an intelligent indoor environment and energy management system for greenhouses. *Energy Conversion andManagement 51*, pages 155–168, 2010.

[Mul13] David J. Mulla. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems Engineering 114 (4)*, pages 358–371, 2013.

[MVS05] Raul Morais, A. Valente e C. Serôdio. A wireless sensor network for smart irrigation and environmental monitoring: A position article. *EFITA/WCCA Joint Congress on IT in Agriculture, Portugal*, pages 845–850, 2005.

[Org12] Modbus Organization. Modbus application protocol specification v1.1b3. disponível em http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf, 2012.

[PH05] D. Puccinelli e M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits and Systems Magazine 5 (3)*, pages 19–31, 2005.

[RSS+09] J.A. López Riquelme, F. Soto, J. Suardíaz, P. Sáncheza, A. Iborraa e J.A. Vera. Wireless sensor networks for precision horticulture in southern spain. *Computers and Electronics in Agriculture 68*, pages 25–35, 2009.

[SEL06] University of Wisconsin-Madison Solar Energy Laboratory. Trnsys a transient simulation program, version 16.1. University of Wisconsin: USA, disponível em http://web.mit.edu/parmstr/Public/Documentation/08-ProgrammersGuide.pdf, 2006.

[SMK13] Abhijit Suprem, Nitaigour Mahalik e Kiseon Kim. A review on application of technology systems, standards and interfaces for agriculture and food sector. *Computer Standards & Interfaces 35 (4)*, pages 355–364, 2013.

[Sri06] Ancha Srinivasan. *Handbook of Precision Agriculture: Principles and Applications*. Haworth Press, First edition, 2006.

[ST94] B.N. Schilit e M.M. Theimer. Disseminating active map information to mobile hosts. *IEEE Networks 8 (5)*, pages 22–32, 1994.

[Sta00] J.V. Stafford. Implementing precision agriculture in the 21st century. *Journal of Agricultural Engineering Research 76*, pages 267–275, 2000.

[STPM05] A. Sheth, K. Tejaswi e et al P. Mehta. Senslide: a sensor network based landslide prediction system. *Proceedings of Sensys '05 — the 3rd International Conference on Embedded Networked Sensor Systems*, pages 280–281, 2005.

[The03] P. S. Thenkabail. Biophysical and yield information for precision farming from near-real-time and historical landsat tm images. *International Journal of Remote Sensing 24*, pages 2879–2904, 2003.

[uRAS08] Aqeel ur Rehman, A.Z. Abbasi e Z.A. Shaikh. Building a smart university using rfid technology. *2008 International Conference on Computer Science and Software Engineering (CSSE 2008), Wuhan, China*, pages 641–644, 2008.

[uRS09] Aqeel ur Rehman e Z. A. Shaikh. *Smart Agriculture*. Bentham Science Publishers Ltd., First edition, 2009.

[WZW06] N. Wang, N. Zhang e H. Wang. Wireless sensors in agriculture and food industry - recent development and future perspective. *Computers and Electronics in Agriculture, Volume 50, No. 1*, pages 1–14, 2006.

[YS04] H. Yanlin e C. Shoulun. Summarization of fertilization model research. *Chinese Journal of Soil Science 35*, pages 493–501, 2004.

[YWHZ13] Xiaoqing Yu, Pute Wu, Wenting Han e Zenglin Zhang. A survey on wireless sensor network infrastructure for agriculture. *Computer Standards & Interfaces 35 (1)*, pages 59–64, 2013.

[ZlYmZ$^+$07] Qian Zhang, Xiang long Yang, Yi ming Zhou, Li ren Wang e Xi shan Guo. A wireless solution for greenhouse monitoring and control system based on zigbee technology. *Journal of Zhejiang University SCIENCE A 8 (10)*, pages 1584–1587, 2007.

[ZWW02] Naiqian Zhang, Maohua Wang e Ning Wang. Precision agriculture: a worldwide overview. *Computers and Electronics in Agriculture 36*, pages 113–132, 2002.

REFERÊNCIAS

Anexo A

Exemplo do código gerado na integração de um atuador

Este anexo contém a classe gerada na integração do equipamento TekOn, demonstrando a capacidade da Framework gerar código através da interface gráfica e mecanismos desenvolvidos. A função "GetTemperature"foi a única que teve de ser alterada de forma a completar os requisitos desta função no que diz respeito ao processamento dos dados recebidos. O resto do código foi gerado automaticamente pela Framework, e a sua fiabilidade foi comprovada no âmbiente de teste descrito na secção 5.3.

A.1 Preâmbulo

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading;
6  using Modbus.Device;
7  using System.IO.Ports;
8
9  public class TekOnAtuador
10 {
11   const int node_offset = 27;
12
13   const int port_ReadTimeout = 5000;
14   const int port_WriteTimeout = 5000;
15
16   const int master_Port_ReadTimeout = 5000;
17   const int master_Port_WriteTimeout = 5000;
18
19   const int master_Transport_Retries = 1;
```

Listing A.1: Preâmbulo

A.2 DefineInterval

```
1  /// <summary>Definir intervalos de envio de temperatura</summary>
2  /// <param name="gatewayDevice">Real slave device number for the ModBus command to
       be delivered. In ModBus RTU this is the gateway number.</param>
3  /// <param name="parameters">Contains the function arguments separated by '_'. If
       this function is directed to a slave device in ModBus RTU, the device number is
        the first element of this string</param>
4  /// <param name="port">Serial port to send ModBus command.</param>
5  /// <param name="semaforo">Semaphor to prevent multiple actions on the serial port
       .</param>
6  /// <returns>Contains the result of the command. If only one string returns, an
       execution error has occurred and it contains the exception message. If contains
        two string, the first contains the state of the command: 'Success' or 'Error'.
        The second string contains the message or data to be handled to the server
       workflow.</returns>
7  public static string[] DefineInterval(int gatewayDevice, string parameters,
       SerialPort port, Semaphore semaforo)
8  {
9    try
10   {
11     bool released = false;
12     bool requested = false;
13
14     IModbusSerialMaster master = start(port);
15
16     master.Transport.WriteTimeout = port_WriteTimeout;
17     master.Transport.WaitToRetryMilliseconds = port_WriteTimeout;
18
19     int registerOffset = 18;
20
21     string[] inputs = parameters.Split('_');
22
23     int slaveDevice = int.Parse(inputs[0]);
24     int interval = int.Parse(inputs[1]); // Intervalo de envio de temperatura
25     int probe = int.Parse(inputs[2]); // Tipo de sonda
26
27     int targetAddress = (slaveDevice * node_offset) + registerOffset;
28
29     List<ushort> modbus_inputs = new List<ushort>();
30
31     ushort interval_low = (ushort)interval;
```

```
32      ushort interval_high;
33      if (interval>65535)
34        interval_high = BitConverter.ToUInt16(BitConverter.GetBytes(interval), 2);
35      else
36        interval_high = 0;
37      modbus_inputs.Add(interval_high);
38      modbus_inputs.Add(interval_low);
39
40      modbus_inputs.Add((ushort)probe);
41
42      byte param1 = (byte)gatewayDevice;
43      ushort param2 = (ushort)targetAddress;
44      ushort[] param3 = modbus_inputs.ToArray();
45
46      try
47      {
48        semaforo.WaitOne();
49        requested = true;
50        master.WriteMultipleRegisters(param1, param2, param3);
51        semaforo.Release();
52        released = true;
53
54        // ***** PARSE YOUR RESULT HERE ***** //
55
56        string data = "success";
57
58        // ****** HANDLE YOUR ERRORS TOO ***** //
59
60        string error = "error";
61
62        return new string[]{ "Success", data };
63
64        // *** In case of error: *** //
65        // return new string[]{ "Custom Error", error };
66      }
67      catch (Exception e)
68      {
69        if (released == false && requested == true)
70          semaforo.Release();
71        return new string[] { "Modbus Error", e.Message.ToString() };
72      }
73    }
74    catch (Exception e)
75    {
76      return new string[] { "Execution Error", e.Message.ToString() };
77    }
78  }
```

Listing A.2: Código gerado relativo à função "DefineInterval”

A.3 GetTemperature

O código presente no excerto A.4 foi substituido pelo código presente entre as linhas 42 e 81 da versão final A.3.

A.3.1 Excerto original

```
1  // ***** PARSE YOUR RESULT HERE ***** //
2
3  string data = "sucess";
4
5  // ****** HANDLE YOUR ERRORS TOO ***** //
6
7  string error = "error";
8
9  // ****** DEFINED ERROR TAGS: ******* //
10
11 // string error = "probe_error";
12 // string error = "sensor_off";
13 // string error = "old_value";
14
15 return new string[] { "Success", data };
16
17 // *** In case of error: *** //
18 // return new string[]{ "Custom Error", error };
```

Listing A.3: Excerto original gerado na função "GetTemperature”

A.3.2 Versão final

```
1  /// <summary>Get Temperature from sensor</summary>
2  /// <param name="gatewayDevice">Real slave device number for the ModBus command to
       be delivered. In ModBus RTU this is the gateway number.</param>
3  /// <param name="parameters">Contains the function arguments separated by '_'. If
       this function is directed to a slave device in ModBus RTU, the device number is
        the first element of this string</param>
4  /// <param name="port">Serial port to send ModBus command.</param>
5  /// <param name="semaforo">Semaphor to prevent multiple actions on the serial port
       .</param>
6  /// <returns>Contains the result of the command. If only one string returns, an
       execution error has occurred and it contains the exception message. If contains
        two string, the first contains the state of the command: 'Success' or 'Error'.
        The second string contains the message or data to be handled to the server
       workflow.</returns>
```

Exemplo do código gerado na integração de um atuador

```
public static string[] GetTemperature(int gatewayDevice, string parameters,
    SerialPort port, Semaphore semaforo)
{
  try
  {
    bool released = false;
    bool requested = false;

    IModbusSerialMaster master = start(port);

    master.Transport.ReadTimeout = master_Port_ReadTimeout;
    master.Transport.WaitToRetryMilliseconds = master_Port_ReadTimeout;

    ushort[] data_modbus; // This array contains the bytes read from the serial
        port
    int bytesToRead = 27;
    int registerOffset = 0;

    string[] inputs = parameters.Split('_');

    int slaveDevice = int.Parse(inputs[0]);
    int requestinterval = int.Parse(inputs[1]);

    int targetAddress = (slaveDevice * node_offset) + registerOffset;

    byte param1 = (byte)gatewayDevice;
    ushort param2 = (ushort)targetAddress;
    ushort param3 = (ushort)bytesToRead;

    try
    {
      semaforo.WaitOne();
      requested = true;
      data_modbus = master.ReadHoldingRegisters(param1, param2, param3); // Your
          result is here
      semaforo.Release();
      released = true;

      // ***** PARSE YOUR RESULT HERE ***** //

      int max_val = 65535;

      int writeTime = data_modbus[18] * max_val + data_modbus[19];
      int liveTime = data_modbus[24] * max_val + data_modbus[25];

      if (liveTime > writeTime + 2 /*error margin of 2 seconds*/)
        return new string[] { "Custom Error", "sensor_off" };

      if (liveTime > requestinterval)
```

```
53          return new string[] { "Custom Error", "old_value" };
54
55        double temperature = 0;
56        if (data_modbus[0] == 0)
57          temperature = ((double)data_modbus[1]) / 100;
58        else
59        {
60          if (data_modbus[0] == max_val)
61            temperature = ((double)(data_modbus[1] - max_val)) / 100;
62          else if (data_modbus[0] >= 0 || data_modbus[0] < 99)
63            temperature = ((max_val * data_modbus[0]) + ((double)(data_modbus[1]))) /
                  100;
64        }
65        string probe_type = data_modbus[20].ToString();
66        string node_model = data_modbus[21].ToString();
67        double batery_voltage = ((double)data_modbus[22]) / 100;
68        string rssi = data_modbus[23].ToString();
69        double board_temperature = data_modbus[26];
70
71        if (board_temperature > 63535)
72          board_temperature = (((double)data_modbus[26]) - 65536) / 100;
73        else
74          board_temperature = ((double)data_modbus[26]) / 100;
75
76        if (data_modbus[0] == 99 || data_modbus[0] == 100)
77          return new string[] { "Custom Error", "probe_error" };
78
79        string data = temperature.ToString() + "|" + probe_type + "|" + node_model +
              "|" + batery_voltage.ToString() + "|" + rssi + "|" + board_temperature.
              ToString();
80
81        return new string[] { "Success", data };
82      }
83      catch (Exception e)
84      {
85        if (released == false && requested == true)
86          semaforo.Release();
87        return new string[] { "Modbus Error", e.Message.ToString() };
88      }
89    }
90    catch (Exception e)
91    {
92      return new string[] { "Execution Error", e.Message.ToString() };
93    }
94  }
```

Listing A.4: Versão final da função "GetTemperature”

A.4 start

```
1  private static IModbusSerialMaster start(SerialPort port)
2  {
3    port.ReadTimeout = port_ReadTimeout;
4    port.WriteTimeout = port_WriteTimeout;
5    port.NewLine = "\n";
6    port.DtrEnable = false;
7    port.RtsEnable = false;
8    IModbusSerialMaster master = ModbusSerialMaster.CreateRtu(port);
9    master.Transport.Retries = master_Transport_Retries;
10   return master;
11 }
```

Listing A.5: Código gerado da função "start”